LA-UR-13-20710
Approved for public release; distribution is unlimited.

Title:	Power-Aware Data Center Project: Final Report
Author(s):	Pakin, Scott D.
Intended for:	Report Web

Power-Aware Data Center Project: Final Report

LOS ALAMOS NATIONAL LABORATORY

PRINCIPAL INVESTIGATOR: SCOTT PAKIN, *pakin@lanl.gov*

31 December 2012

Abstract

Due to cost and space concerns, data centers are becoming increasingly limited in their ability to enhance their power infrastructure to support increased compute power on the machine-room floor. The key hypothesis underlying this work is that there is a substantial amount of "trapped capacity" in existing supercomputing data centers. That is, more power infrastructure may be allocated to existing supercomputers than these supercomputers (a) can possibly or (b) typically draw. Consequently, it may be possible to increase the amount of computational throughput delivered using an existing power infrastructure as long as power can be capped at a lower-than-peak level.

This study comprises two parts. The first part is to quantify the amount of trapped power capacity within a major data center, and the second part is to exploit this trapped capacity either to reduce recurring and infrastructure costs or to utilize more computing resources for a given cost. Our trapped-capacity studies were carried out on production supercomputers at Los Alamos National Laboratory (LANL)'s main supercomputing data center and represent, to our knowledge, the largest study of its kind in terms of aggregate performance, aggregate power, and length of time. Our power-capping work was performed on a testbed version of one of the production supercomputers at 1/10th scale of the production supercomputer.

Our findings from the trapped-capacity study are that LANL supercomputers contain significant trapped capacity, not just on average but even in the worst case; variability in power draw can be quite different across different architectures; there is a qualitative difference between power drawn while running benchmarks and when running a production workload; power capping has the potential to free large amounts of power and cooling infrastructure with minimal impact on applications; an ability to coschedule high-power-consuming jobs with low-power-consuming jobs would offer the potential to reduce peak power draw to further support penalty-free power capping; and that "race to halt" appears to be a more effective way to run a scientific workload than are various power-saving strategies.

Our experience with implementing a cluster-wide power-capping scheme lead us to the following conclusions: without hardware support for cluster-wide power capping, a stochastic approach is essential to compensate for the power-cap mechanism's slow response time relative to an application's ability to suddenly increase its power consumption; numerous policy decisions must be made to determine how power capping should be imposed; and that implementing power capping can be straightforward, but there are myriad small engineering challenges to overcome.

Contents

CHAPTER 1

Executive summary

Los Alamos National Laboratory (LANL) was responsible for two key components of the Power-Aware Data Center (PADC) project: quantifying the amount of trapped power capacity in LANL's production supercomputing data centers and devising a scheme to exploit this trapped capacity by capping power, thereby freeing up power infrastructure for use by additional supercomputing capacity. The following are the main takeaway points from our findings.

LANL's main data center contains many megawatts of trapped capacity. The exact amounts depend on the precise definition of trapped capacity used (capacity over the maximum power ever observed, capacity over LINPACK power, capacity over some power quantile, etc.). However, even going by the most conservative calculation—maximum observed power—LANL's main data center contains at least 3.6 MW of trapped capacity: 2.3 MW for the Roadrunner supercomputer, 1.1 MW for the Cielo supercomputer, and 0.2 MW for the Luna supercomputer. More trapped capacity likely exists, but our study focused on these three supercomputers as they are LANL's largest.

Trapped capacity is partly due to procurement constraints. Supercomputer power is budgeted long before the supercomputer is installed to give the facilities engineers time to prepare the data center for the new procurement. Nameplate power therefore represents the supercomputer vendor's best estimate for the maximum power that the supercomputer will draw when running LINPACK. This use of estimation leads to a large discrepancy between the power a supercomputer requires and the power that is delivered to it.

Trapped capacity is partly due to workload characteristics. LINPACK is used as LANL's power benchmark, but our data indicate that LINPACK is unrepresentative of LANL's scientific computing workload. According to our measurements, applications typically consume only about 70–75% of LINPACK power.

Power can be capped with minimal application impact. Such a small fraction of LANL's workload is power-hungry that power can be capped quite substantially with minimal impact on applications. For example, if Cielo's power were capped at 67% of its nameplate power, only 1% of the 278,394 power samples we took over the course of 16 months would represent windows of time impacted by that cap.

System-wide power-capping requires more than per-node power caps. Per-node hardware power caps can be expected to work reasonably well for throughput workloads, in which each node runs an independent set of jobs. However, per-node power caps, while sufficient to impose a global power cap, will unnecessarily degrade the performance of tightly synchronized, parallel applications that run across many nodes. Such applications run only as fast as their slowest process. Hence, throttling the performance of a single node can reduce the performance of the entire application. In addition, in a system-wide power-capping scheme, it is acceptable for one node to run above the power cap as long as other nodes are running below the cap, and this should be exploited to maximize performance subject to global power constraints.

Power capping should be aware of job scheduling. Because the performance of a parallel application is limited by the performance of its slowest constituent process, CPU frequencies should be raised or lowered on a per-job rather than a per-processor, per-node, or per-system basis. Otherwise, high-power/high-performance processes will stall waiting to synchronize with low-power/low-performance processes, and this will delay the entire execution and waste some of the power budget that could have been applied elsewhere. For a power-capping scheme to know which jobs are running on which nodes, it needs to be integrated with the parallel job scheduler.

System-wide power capping requires unachievable response times. A supercomputer can swing from minimal to maximal power consumption on the order of milliseconds. Software cannot practically throttle CPU frequency (the main controller available for power capping) fast enough to account for sudden increases in power. It is even questionable whether, at supercomputer scales, even hardware could successfully detect power swings and adjust CPU frequencies accordingly.

Statistics can compensate for inherently slow response times. We investigated using statistical analyses to perform short-term predictions of how much power will be consumed globally based on prior measurements. We conclude that this is likely to be a practical approach to system-wide power capping at supercomputer scales.

Policy decisions cannot be avoided. One role of a system-wide power-capping scheme is to select jobs whose CPU frequencies should be reduced to ensure that the global power cap is not exceeded. This selection can be based only on policy; there is no inherently optimal way to select victim jobs. Each alternative has its own strengths and weaknesses, and only a human can make the judgment call as to what is best for that supercomputer's users. Consequently, any power-capping implementation should be sufficiently flexible as to support a broad scope of possible policies.

CHAPTER 2

Project overview

The following text is copied essentially verbatim from the Power-Aware Data Center (PADC) project's statement of work. The purpose is to set the stage for the remainder of this report. Each subsequent chapter likewise begins with an excerpt from the statement of work as a form of introduction.

Introduction

Work to be divided between 2 labs
Labs to collaborate with each other, and with the sponsor
Labs will each have one shared task area, and one individual task area.

Overview

The objective of this research effort is to explore implementing intelligent load-side control of data center equipment to gain efficiency and/or additional capacity. The load-side control techniques to be explored will include power capping of individual loads, throttling power supply to systems during periods when maximum computational speed is not required, and coordinating load profiles of systems to avoid overlapping of individual system demand peaks.

Problem Statement

Due to the increasing power density of today's computing systems, data centers are becoming power and cooling limited, which means that they have reached a point where the full amount of raised floor space cannot be populated with equipment because the power and cooling infrastructure is already operating at full capacity. Adding additional power and cooling infrastructure requires significant time and investment, and is typically very disruptive to operational computing systems. Even when the infrastructure is operating at full capacity, there is some margin between actual system demand and available power and cooling capacity. Some of this margin is required for safety and reliability; however, some of the margin is created when systems are not operating at full nameplate rating. It is this latter component of margin, which will be referred to as trapped capacity, which this research effort seeks to exploit.

Trapped capacity is the difference between the infrastructure capacity allocated to a given system and the actual peak demand of that system. For example, the electrical feeder to a rack of servers is typically sized to be able to feed all of the servers running simultaneously at maximum power draw. However, in normal operation the peak electrical demand of the rack may never exceed half of the demand that was used to size the feeder. Yet the excess capacity is held in reserve "just in case" the worst case demand was to occur. This reserve excess capacity is referred to as trapped capacity.

Every computing system has a unique power duty cycle. This duty cycle reflects the fact that systems normally do not run at full power consumption rates all of the time. Instead, power demand rises and falls depending on the specifics of the computation being performed. For example, power consumption may be high during periods of heavy computation and lower while results are being written to memory. There may be times when throttling the power demand by slowing down, or turning off certain subsystems may have very little effect on the overall performance of the system, while significantly reducing power demand.

Load profile coordination considers ways to choreograph system load profiles to avoid concurrence of individual system load peaks. If successful, this could allow the same amount of work to be performed at a lower total system peak demand, which in turn would reduce the amount of required infrastructure.

The idea for this research is to place power caps on servers/racks to force systems and/or programmers to operate within a power limit to reduce overall data center demand or create demand margin that could be used to power additional systems. The caps would be determined and enforced by an overarching power management system that would coordinate available power with factors such as total data center demand, job/system priority, job completion time criticality, etc.

The goals for the envisioned system include:

- Regain "trapped capacity" to fully utilize data center power and cooling infrastructure. Minimize infrastructure "over provisioning"
- Given a power and cooling constrained data center, implement power usage control through dynamic "power knobs" so that "idle" or "inefficient" cycles can be reclaimed and used to support additional mission capabilities on otherwise unused floor space.
- Actively control the amount of power or energy allotted to each rack or system on a data center floor for the purpose of improving energy efficiency.

CHAPTER 3

Trapped capacity

The following text is copied essentially verbatim from the PADC project's statement of work.

The first task is to identify and quantify trapped capacity in existing data centers. Trapped capacity is the difference between the infrastructure capacity allocated to a given system and the actual peak demand of that system. For example, the electrical feeder to a rack of servers is typically sized to be able to feed all of the servers running simultaneously at maximum power draw. However, in normal operation the peak electrical demand of the rack may never exceed half of the demand that was used to size the feeder. Yet the excess capacity is held in reserve "just in case" the worst case demand was to occur. This reserve excess capacity is referred to as trapped capacity.

One of the goals of this research is to identify trapped capacity in an existing data center. This can be done by measuring and monitoring individual system loads over time to establish load profiles for each system in the data center. Once established, these load profiles can be compared to circuit sizing to determine the amount of trapped infrastructure capacity.

Tasks

- Each Lab will perform the necessary measurements and analysis to determine the trapped capacity in one or more of their data centers.
- Each lab will document the procedures used to:
    1. Determine the amount of infrastructure allocated/reserved for each system in their data center.
    2. Measure and record power demand for each system.
    3. Establish time-based load profiles for each system.
    4. Determine the amount of trapped capacity that could be recovered at the system level, and at the data center level.

Milestones

1. Identify all systems in the data center and the amount of power infrastructure allocated to each. (x months)

2. Measure/record/document power demand data for each system. ($+x$ mos.)
3. Develop typical load profiles for each system. ($+x$ mos.)
4. Produce report that:
   a) Documents methodology used to generate load profiles
   b) Compares load profiles to infrastructure allocations, and quantifies the amount of infrastructure that could be reclaimed through load management techniques.

3.1 Introduction

A major challenge of exascale computing is to deliver a thousandfold increase in performance while only slightly increasing power consumption over current petascale systems [8, 25]. While there has been much research on increasing power efficiency within various components of a supercomputing system—processors, interconnection networks, system software, programming models, algorithms, and applications—and numerous controlled studies, there is comparatively little data available describing how much power a supercomputer draws while running a production scientific workload. The goal of this part of the PADC study is to fill that gap by presenting power data measured at the main supercomputer data center at Los Alamos National Laboratory (LANL) and on three of the world's fastest supercomputers (based on the semiannual Top500 list of supercomputer performance [29]).

We present full-system power data measured since inception of two production supercomputers, Cielo and Roadrunner, and one preproduction supercomputer, Luna. (The three machines are described in detail on pages 8–12.) Combined, these systems represent a total of just under 13,500 nodes, making this one of the largest studies of power drawn while running a real workload. For Luna, we further include some controlled studies to analyze the extremes of that system's power usage and examine the discrepancies between measuring power at the switchboard level and at the sub-rack level.

Our findings are that power variability differs substantially across architectures; from a power perspective, real scientific workloads bear little in common with the LINPACK benchmark (not too surprisingly); the difference between worst-case and average-case power draws indicates that supercomputing data centers may contain a fair amount of "trapped capacity" in their power systems, more if power capping can be implemented on a full-system basis; job schedulers theoretically have the potential to increase trapped capacity even further by pairing jobs of different power envelopes; and energy savings are unlikely to be achieved merely by frequency and voltage scaling, given how supercomputers are currently run.

We anticipate that this report will assist future power studies that require knowledge of real-world supercomputer power data to drive their approach and solutions.

The remainder of this chapter is organized as follows. We discuss the most relevant pieces of related work in Section 3.2. Section 3.3 describes LANL's data center in terms of its power characteristics and the main supercomputers it hosts. The main section of the chapter is Section 3.4, where we present our power measurements and associated analyses. Section 3.5 helps explain our findings through the use of additional power studies in a more controlled environment. Section 3.6 briefly describes some prospects for follow-on research. Finally, we draw some conclusions from our trapped-capacity study in Section 3.7.

3.2 Related work

Fan, Weber, and Barroso quantify the power usage of three workloads that run at one of Google's large data centers: Google Web search, GMail, and various offline MapReduce jobs [9]. As in our work, they examine power characteristics at the rack, sub-cluster, and full-cluster levels. The key characteristic that distinguishes our work from theirs is that we focus on a production scientific workload in which

applications tend to be more tightly coupled than search and email services and MapReduce jobs. Scientific applications generally include substantial communication within sets of processes/nodes, and the workload as a whole tends to allocate and free large numbers of nodes at once. This can incur sudden power changes and therefore requires great care in implementing power capping. Also, unlike Google's large data centers, LANL's supercomputers run at fairly constant job load over time, limiting the usefulness of the Google paper's studies of reducing power during off-peak times. LANL's data center has no off-peak times. Although incidental, our paper presents data that covers twice as many nodes and for twice the duration as Fan, Weber, and Barroso's work.

In the context of scientific computing, Laros et al. have performed a number of large-scale power studies on two Cray XT supercomputers [26]. They quantified the power usage of a number of representative scientific applications with different CPU frequencies and also when scaling back communication bandwidth to reduce the power drawn by the network. The main difference between our work and Laros et al.'s is that they performed controlled studies of individual applications while our study represents a production workload with many applications of different sizes running concurrently in a space-shared manner across entire systems.

Hennecke et al. discuss power measuring and monitoring on the large, production Blue Gene/P supercomputer at the Jülich Research Center [21]. Two areas of comparison with our work are Hennecke et al.'s presentation of a histogram of power consumption while running a production workload over a long period of time and their graphs of power usage when performing a controlled, single-node application study. Our work, in addition to incorporating more analysis of production-workload data and including a controlled application study involving a substantial number of nodes, also examines three top supercomputers for over four times the duration of Hennecke et al.'s work.

3.3 Facility

3.3.1 Power infrastructure

Figure 3.1 illustrates the power-monitoring infrastructure used in LANL's main data center. 13,200V enter the data center and are converted to 3-phase, 480V power to feed the substations. The substations transmit power through a rotary uninterruptible power supply (RUPS) to a number of switchboards, each of which feeds multiple power distribution units (PDUs) on the machine-room floor. These convert the power into 3-phase, 208V or 480V power for distribution to the compute racks. An assumption the facilities engineers make is that no PDU will ever exceed 80% load (e.g., 200 kVA for a 240 kVA PDU). Otherwise, the unit runs the risk of observing a voltage sag, tripping the circuit breaker, cutting power to the associated racks, and leading to costly system downtime.

The data center can deliver an aggregate of 19.2 MW of power. From January through April 2012, the facility has generally been running at about half of peak capacity and has averaged a power usage effectiveness (PUE)—the ratio of total power to IT equipment power [36]—of 1.41, with facilities measured at the substation and IT measured at the switchboard. A PUE of 1.41, while far from the state of the art in efficient data-center design, is nevertheless competitive with that of other large-scale data centers and is substantially better than the 1.86 averaged by the data centers in Greenberg et al.'s study [17].

Power can be monitored at each switchboard and at each rack. In fact, two of our supercomputers each provide finer than rack-resolution monitoring: Roadrunner supports intra-node monitoring, and Luna supports monitoring at the "shelf" (10-node) level. Switchboard monitoring is automatically logged and stored. Originally, logging was performed at 15-minute intervals. In late February/early March 2012 we increased the logging rate to 1-minute intervals to help correlate the switchboard readings with other power monitors. Currently, rack and sub-rack monitoring is done only on demand, by explicitly polling the monitoring devices. We therefore have comparatively little data at this level.

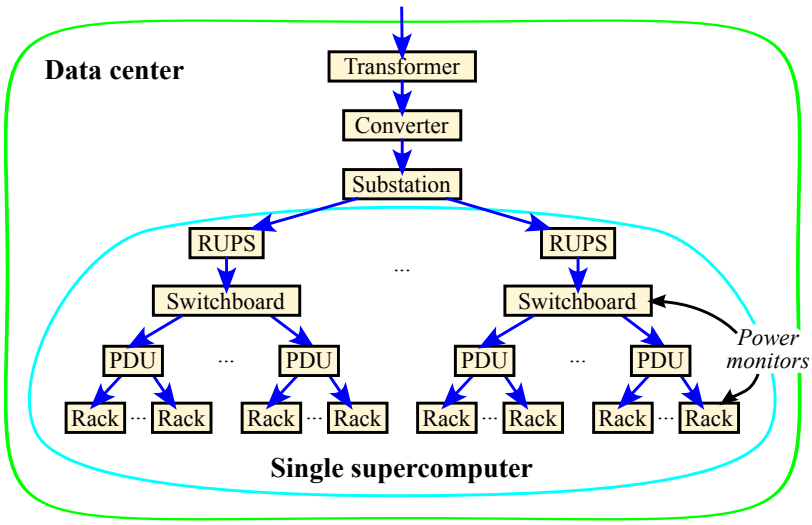


Figure 3.1: Power-monitoring infrastructure

An important feature of the way LANL's data center is configured is that each switchboard is associated with a single supercomputer. Consequently, we can measure power independently for each supercomputer, a capability that would not otherwise be possible. Note, however, that cooling and storage (parallel filesystems) are not associated with any particular supercomputer and reside on separate switchboards. Figure 3.2 presents a screenshot of Geist's Environet system monitoring one of the Cielo supercomputer's six switchboards. Although all of the meters shown with an adjacent triangle icon are logged, in this study we rely only on the historical wattage data.

In addition to power, the cooling infrastructure exacts substantial recurring costs. Cooling for the data center housing Roadrunner, Cielo, and Luna draws 2.8 MW of power at a cost of about US$1M/MW/year.

3.3.2 Supercomputers studied

Table 3.1 lists some key characteristics of the systems we used in our study. The "Top500 rank" column represents each supercomputer's performance ranking based on the November 2012 Top500 list [29], where "1" indicates the world's fastest supercomputer according to that metric. The "Max. power/rack (kW)" and "Max. power/system (MW)" columns represent each rack or system's "nameplate" power—the maximum power the rack or entire system can draw. Section 3.3.3 explains how nameplate power is determined. The nameplate power, plus an additional safety margin (Section 3.3.1), is the total power allocated to the system. The "LINPACK (MW)" column represents the system power drawn while running the High-Performance LINPACK (HPL) benchmark [7]. Finally, the "Idle (MW)" column represents the system power drawn while idle. Idle-power values were computed statistically from empirical data as explained in Section 3.4.4.

Roadrunner was the world's first petascale system [4]. It employs a hybrid architecture with AMD Opteron CPUs and IBM Cell processors as computational accelerators. Each node comprises a total of 40 cores: two sockets of dual-core Opteron and four sockets of Cell, with each Cell socket containing one general-purpose control processor and eight vector processors [23]. Nodes are connected by a dual-data-rate (DDR) InfiniBand [22] fat tree from Mellanox. Although the Cell processor is now defunct, Roadrunner is nevertheless representative of contemporary accelerated supercomputers—such as those built with GPUs or MICs—in that it it requires separate programming of the host and accelerator; there are multiple memory-address spaces per node; communication between the host and accelerator is expensive;

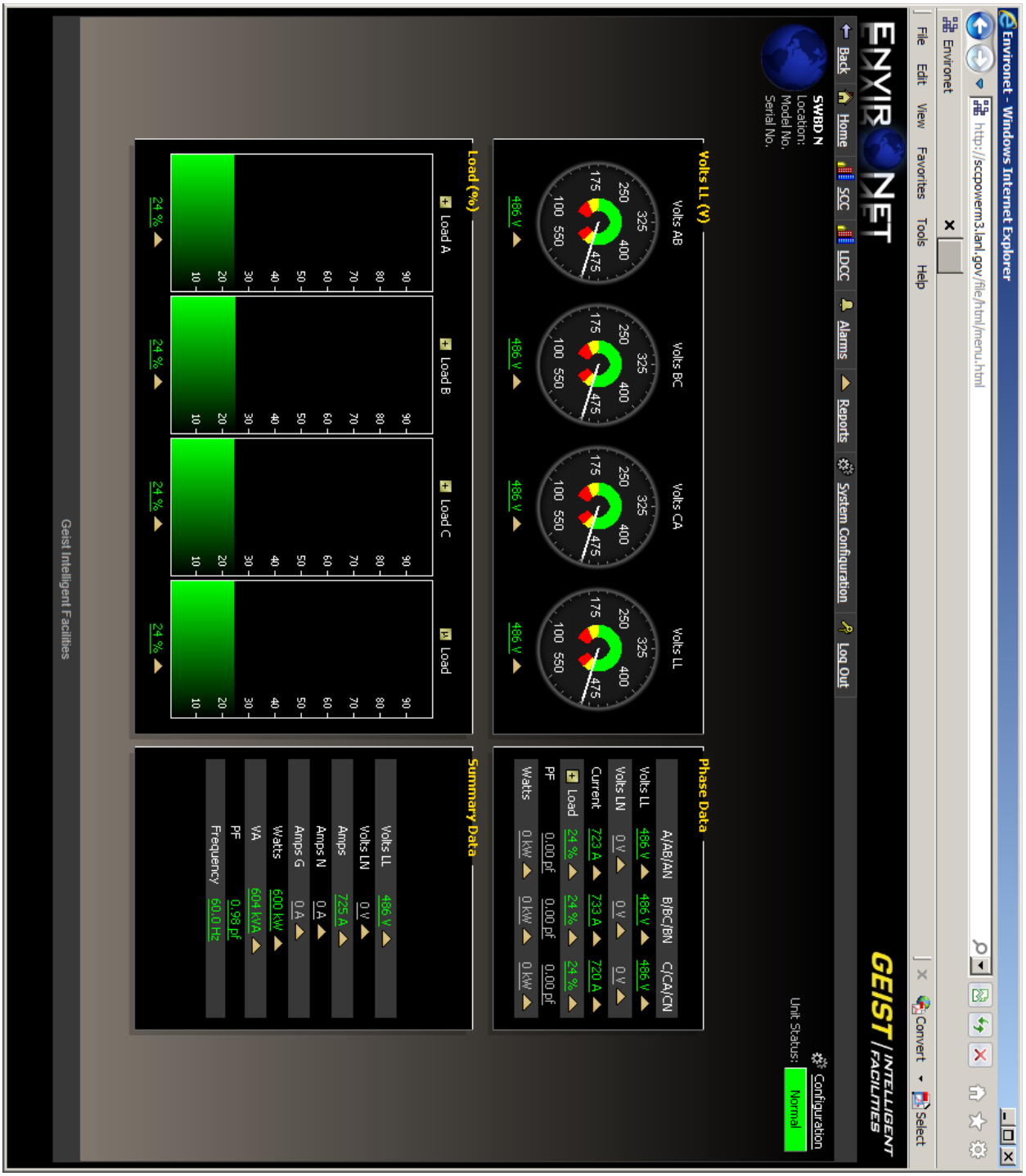


Figure 3.2: Switchboard-level power monitoring

Table 3.1: Supercomputers used in this study

Machine	Top500 rank	Switch-boards	Racks	Nodes	Cores	Max. power/ rack (kW)	Max. power/ system (MW)	LINPACK (MW)	Idle (MW)
Roadrunner	22	4	272	3,060	122,400	15	4.08	2.35	1.56
Cielo	18	5	96	8,892	142,272	54	5.18	3.98	1.84
Luna	64	1	35	1,540	24,640	24	0.84	0.54	0.21
Total	—	10	403	13,492	289,312	—	10.10	6.78	3.61

the accelerator must employ the host's assistance for inter-node communication; and the overall system is quite power-efficient. In fact, due to the Cell's high peak performance per watt, Roadrunner ranked sixth on the second Green500 list [12] even though its aggregate performance was substantially higher than virtually every other system on the list.

Cielo is a large Cray XE6 system [44]. Each node contains two sockets of 8-core AMD Magny-Cours CPUs, and nodes are connected with a Cray Gemini network [1] organized as a 3-D torus. What makes Cielo interesting from a power perspective is that the communication fabric is integrated into the nodes, not separated as it is in the other two systems in Table 3.1. Consequently, racks are more homogeneous in Cielo, while Roadrunner and Luna incorporate network switches within a subset of the racks. For example, Roadrunner's first-level InfiniBand switches are housed in 17 out of its 272 compute racks, and there are an additional 4 second-level InfiniBand switches separate from the compute racks but on a shared switchboard.

Luna is a significantly different system from both Roadrunner and Cielo. First, it is a commodity cluster with no custom hardware [10]. Each node contains two sockets of 8-core Intel Sandy Bridge CPUs, and nodes are connected with a quad-data-rate (QDR) InfiniBand [22] fat tree from QLogic. Second, Luna is intended to run a large number of small jobs (tens to hundreds of nodes) rather than a small number of large jobs (thousands of nodes), as is the case for Roadrunner and Cielo. Finally, while Roadrunner and Cielo are running a production workload, at the time we gathered our data, Luna had not yet stabilized to the point where general users were allowed onto the system. (It has since joined Roadrunner and Cielo as a production resource.) Because of Luna's inchoate status, it was comparatively easy to reserve the entire machine for our work; our controlled studies are therefore performed exclusively on Luna.

Figure 3.3 shows the layout of the machine-room floor. Cielo is the square bounded by F8–G9 in the diagram. Roadrunner is the large rectangle centered within C10–H12. And Luna lies on the opposite side of the machine room from Roadrunner, appromiately from D3–E4.

Of the three supercomputers used in our study, Roadrunner runs the smallest variety of applications, and Cielo runs the largest. Roadrunner and Luna run primarily LANL-developed applications, while Cielo additionally runs applications from other U.S. Department of Energy laboratories. Roadrunner, Cielo, and Luna are used almost exclusively for large-scale scientific simulations that ensure the safety, security, reliability, and performance of the U.S. nuclear-weapons stockpile without nuclear explosive testing [31]. The physics being simulated may include combinations of hydrodynamics, radiation transport, fission, thermonuclear burn, high-explosives burn, and/or instabilities and mix [2].

Simulating macroscopic phenomena at the subatomic level with high fidelity leads to an unsatiable demand for computing resources. It is not uncommon for simulations[1] to run for long periods of time (often months of wall-clock time) on large numbers of processors and perform only occasional I/O (a

[1] In this context, we distinguish "simulation" from "job". Time limits imposed on jobs restrict execution to 10–24 hours of wall-clock time (depending on queue, machine, user group, etc.) Hence, a single simulation may comprise a substantial number of jobs, each one picking up via a restart file where the previous job left off.

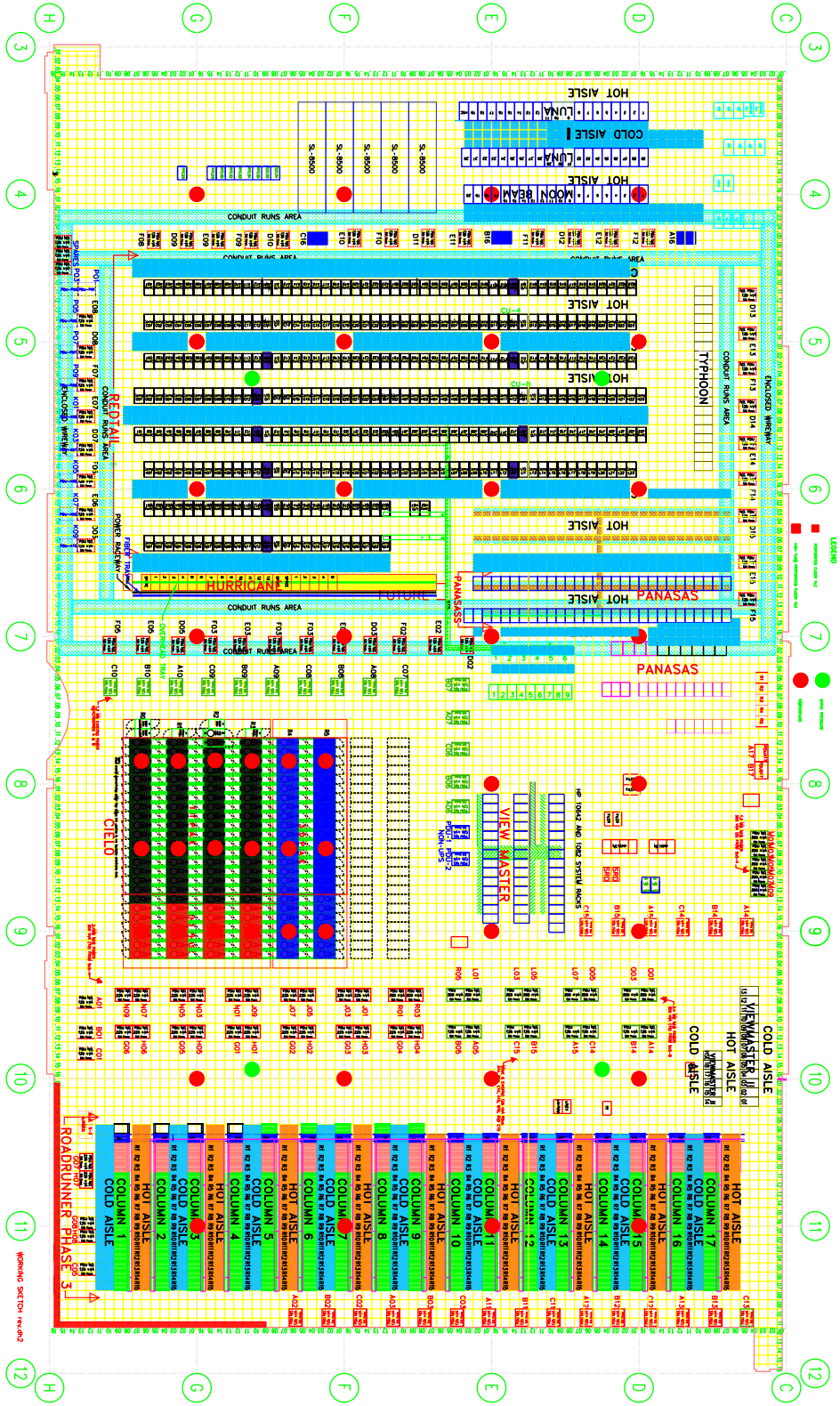


Figure 3.3: Layout of the machine-room floor

few minutes every few hours) for checkpointing or writing out results [37]. In addition to large, scientific simulations, LANL supercomputers host a number of single-node jobs such as compilations, debugging sessions, interactive usage, and other tasks. Furthermore, it is common practice to run a large number of modest-sized parameter sweeps (tens of nodes) to identify likely parameters of interest then feed those parameters into a few long-running simulations that occupy many hundreds or thousands of nodes. In short, the workload running on LANL supercomputers is qualitatively different from that running on, say, Google's [9].

3.3.3 Allocation of power

The "Max. power/system (MW)" column in Table 3.1 merits some explanation. It takes many months to upgrade a data center's power and cooling infrastructure to support a new, power-hungry supercomputer, and this naturally needs to precede supercomputer installation. For example, the project schedule shown in Figure 3.4 indicates that the design of the facility upgrade for LANL's Trinity supercomputer, to be delivered in FY2015, began in April 2012 and that construction will commence in April 2013. At the time of this writing, December 2012, the technical requirements have not yet been finalized, and the request for proposals has not yet been delivered to vendors. While Trinity is expected to require a more-substantial-than-normal facility upgrade, this does demonstrate that power infrastructure must be budgeted to a supercomputer long before the supercomputer exists. This is especially true of top-tier supercomputers that utilize novel hardware, which may not be available in its final form until shortly before supercomputer delivery.

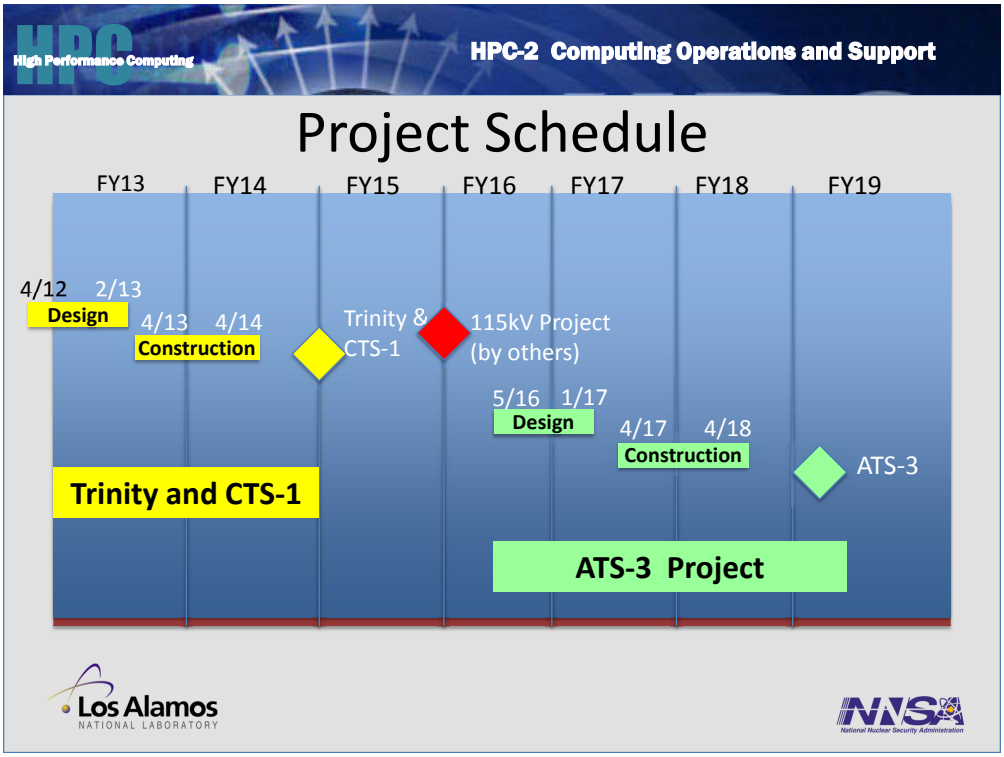


Figure 3.4: Facility upgrade schedule for two future LANL supercomputers

As a consequence of this discrepancy between the time of power-infrastructure installation and that of supercomputer installation, vendors are asked to provide their best estimate for the full system's

worst-case power draw. This estimate, listed as "Max. power/system (MW)" in Table 3.1, is used to allocate infrastructure to the supercomputer. We quantify the accuracy of these estimates in Section 3.4.2.

3.4 Production workloads

In this section we analyze the power drawn by three supercomputers over a 16-month period while performing their normal workloads. We begin by presenting the power drawn over time for each of Roadrunner, Cielo, and Luna. Figure 3.5 shows measured power in kilowatts over a date range from the start of 2011 to the end of April 2012 for the three systems, and Table 3.2 summarizes the data. The following are some points of clarification:

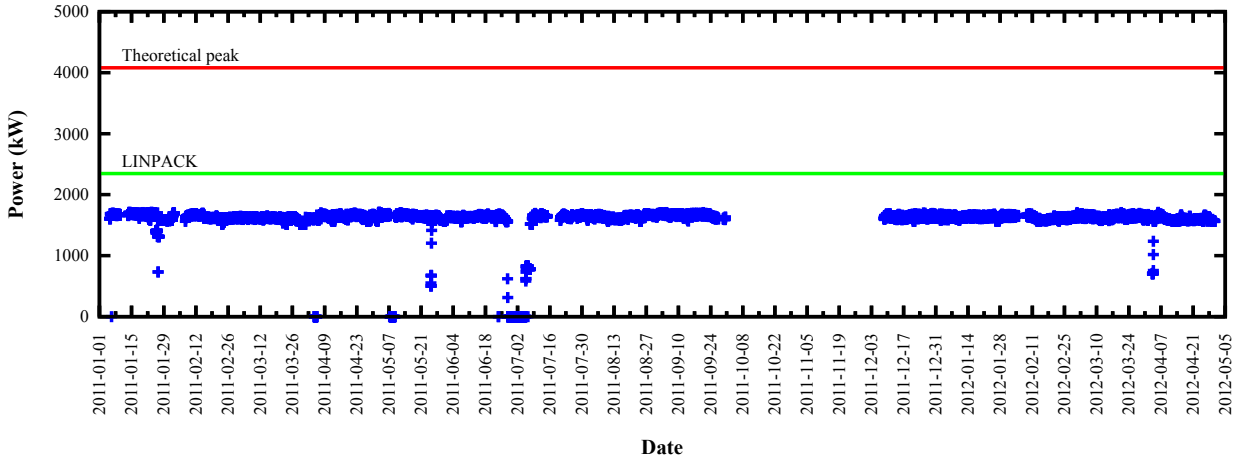
- The *y* axis varies across the subfigures to show more clearly the measured power relative to each machine's theoretical peak power and the power it consumed while running the LINPACK benchmark, which is the metric that orders the Top500 list of supercomputer performance [7].
- The power plateaus in the Cielo data (Figure 3.5(b)) correspond to increases in the number of nodes in the system.
- At the time of our measurements, Luna's LINPACK power had not yet been reported for the complete system. Figure 3.5(c) and Table 3.1 extrapolate the complete-system LINPACK power from the June 2012 Top500 list.
- Luna came online more recently than the other two systems so its power data extends over a lesser range.
- The small gap in data from the end of June to the beginning of July 2011 represents a complete machine-room shutdown as a precautious measure due to a major wildfire coming dangerously close to the Laboratory [42].
- The large gap in data from October to November is due to an upgrade of the switchboard-monitoring software that resulted in some data loss.

Table 3.2: Descriptive statistics of supercomputer power usage

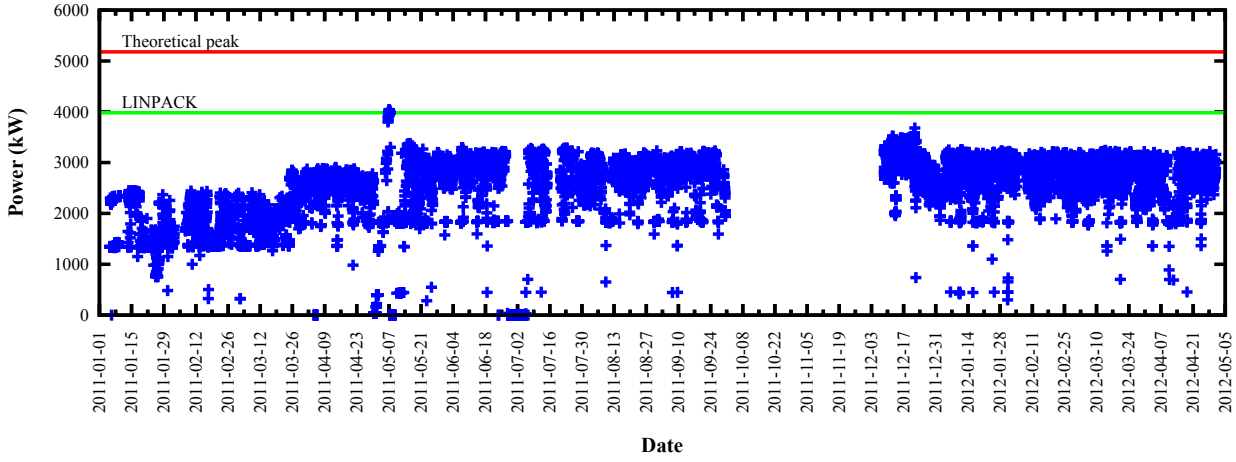
Statistic	Roadrunner (kW)	Cielo (kW)	Luna (kW)
Maximum	1,733	4,043	682
Median	1,632	2,988	352
Mean (μ)	1,623	2,839	327
Std. dev. (σ)	110	555	126

3.4.1 Initial observations

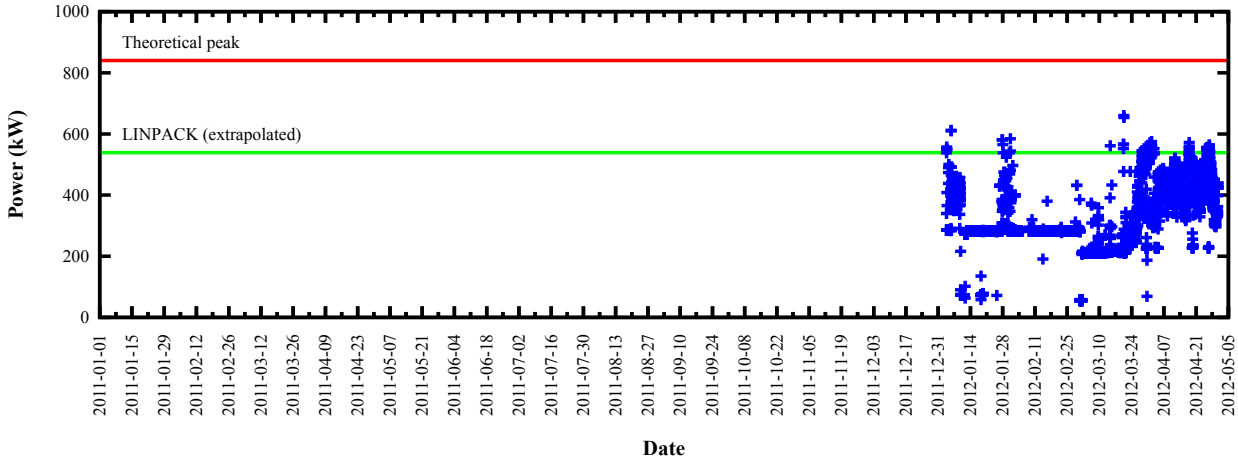
The following are some observations one can make from the data shown in Figure 3.5 and Table 3.2. First, Roadrunner has the most consistent power draw of the three supercomputers, with a relative standard deviation of only 6.8% versus almost 20% for Cielo and almost 40% for Luna. We believe this is due to a lack of clock or power gating in the version of the Cell processor used in Roadrunner [18]. That is, the Cell processors are always running at full power even when idle, and their power draw represents a substantial



(a) Roadrunner



(b) Cielo



(c) Luna

Figure 3.5: Power draw over time of three LANL supercomputers

fraction of the system total. In contrast, although Cielo and Luna do not run a dynamic power governor that raises and lowers clock speeds on demand [32]) and are instead configured always to run at the highest available clock frequency, both the Magny-Cours and Sandy Bridge processors have the ability to aggressively reduce power consumption when idle using techniques such as clock and power gating. Furthermore, buses and other system components draw less power when idle than when transferring data.

To test the hypothesis that power consumption varies even in the absence of frequency scaling, we measured the power consumption on a Luna-like system, Caddy, both when idle and when running a parallel application (xRAGE [16]) across all cores on all nodes. In both cases, the CPU clock frequency was hard-wired to the maximum rate, 2.6 GHz, and Turbo Mode [15] was enabled, which produced an effective CPU frequency of 3.0 GHz. As Figure 3.6 indicates, we observed an average of 147 W per node when idle and 336 W per node when running the application—a factor of 2.3. Excluding the contribution from the CPU sockets (i.e., just considering memory, I/O buses, I/O devices, etc.), the remaining node power increased by a factor of 1.4, from 113 W to 160 W. Figure 3.6 also demonstrates that power consumption at idle is independent of clock frequency while power consumption during application execution increases with clock frequency (roughly linearly, according to other experiments we performed). In short, node power consumption varies even without frequency scaling, and this variation is not due solely to the power consumed by the processors.

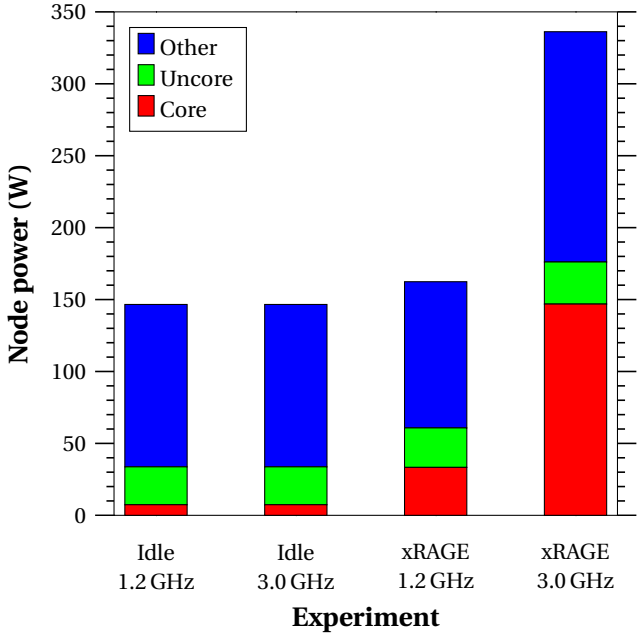


Figure 3.6: Power consumption by CPU frequency when idle or computing

To determine if Cielo's power variability is localized or spread across the entire system, we plotted the individual power contribution of each of Cielo's five switchboards. As Figure 3.7 indicates, all five switchboards observe similar fluctuations in power draw. Not all switchboards service the same number of nodes, which is why the bottom two curves (magenta and red) exhibit different average power draws from the top three curves (green, blue, and cyan), which are largely coincident.

The second observation one can make from Figure 3.5 and Table 3.2 is that the scientific workload running on Roadrunner draws only 69.4% of the power that LINPACK draws, and the scientific workload running on Cielo draws only 71.3% of LINPACK power. These numbers indicate that from a power perspec-

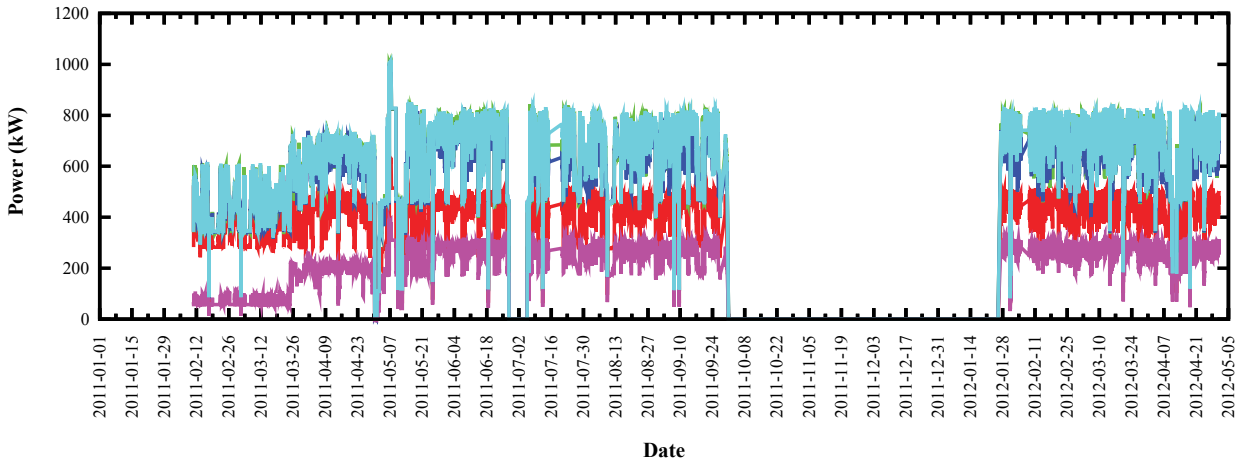


Figure 3.7: Per-switchboard power data for each of Cielo's five switchboards

tive, LINPACK is not particularly representative of the scientific workload normally executed at LANL. As a third-party comparison, Hennecke et al. report that JUGENE, the large Blue Gene/P supercomputer at the Jülich Research Center, draws 29 kW/rack when running LINPACK but an average of only 23 kW/rack (79.3% of LINPACK) when running applications [21].

One difference between applications and LINPACK that may explain the discrepancy in power usage is that applications tend to spend more time than LINPACK blocked on communication and I/O, during which time they draw less power than when stressing the CPU and memory system. In contrast, LINPACK is compute-bound, performing $O(n^3)$ floating-point operations for only $O(n^2)$ memory and communication operations [20].

As a third observation, one can quantify what is sometimes referred to as a supercomputer's "trapped capacity"—the difference between the infrastructure capacity allocated to a given system and the actual peak demand of that system. Going by the maximum power draw ever observed in the time frame represented by the data, one can consider Roadrunner to have 2,347 kW (58%), Cielo to have 1,141 kW (22%), and Luna to have 158 kW (19%) of trapped capacity. In practice, however, facilities engineers typically include a safety margin between the maximum power a system is expected to draw and the power available to that system to reduce the risk of tripping a circuit breaker, as mentioned in Section 3.3.

We now examine our data center's trapped capacity in greater detail.

3.4.2 Trapped capacity

Table 3.3 quantifies the amount of trapped capacity available in LANL's data center. It utilizes the maximum power listed in Table 3.1 and considers different definitions of peak system demand including the measured LINPACK value and the maximum power draw observed. Note, however, that these two values may not be distinct; either some applications do in fact have similar power characteristics to LINPACK or, more likely, our measurements include a few LINPACK runs.

Trapped capacity can be increased even further if one assumes the ability to explicitly limit the power a system draws by throttling frequency and voltage parameters, for example by using Intel's Node Manager [19]. Table 3.3 therefore also lists the available trapped capacity if power is capped at the 75th, 95th, and 99th percentiles, i.e., the power levels such that 75%, 95%, or 99% of the workload runs at full speed. The percentiles listed in Table 3.3 are obtained using the empirical percentiles from the more than 100,000 data points represented by Figure 3.5.

For example, Table 3.1 lists Luna's maximum power as 0.84 MW. If we require 95% of the data points in Figure 3.5(c) to lie below a power cap, then a data-quantile analysis indicates that the power cap must be set no lower than 507 kW. This leaves Luna with 804 kW – 507 kW = 333 kW (40% of 804 kW) of trapped capacity, which is what is listed in Table 3.3.

The "100%" row in Table 3.3 indicates that there is a large potential for power improvements on all three systems. By "power improvements" we mean that more racks can be added without having to upgrade the facility's power infrastructure. Table 3.3 also indicates that if power capping is utilized (which, according to Laros et al.'s study, will degrade the performance of some applications more than others [26]), further power improvements are possible. Even with 99% of the workload unaffected, a power cap can recover 33–59% of the system's fixed power costs. Recall from Section 3.3.3, however, that the maximum power represents the vendor's best estimate, made far in advance of system installation. In the case of Roadrunner, the novelty of the hardware and the lack of an existing hybrid Opteron/Cell LINPACK implementation contributed to the inaccuracy of the estimate, which is why more power improvements may be possible on Roadrunner than on the other two supercomputers.

Table 3.3: Trapped capacity relative to various power maxima, using data quantiles with zeros removed

Assumed % of max. power	Roadrunner (kW)	Cielo (kW)	Luna (kW)
LINPACK	1,730 (42%)	1,204 (23%)	300 (36%)
75%	2,431 (60%)	1,962 (38%)	417 (50%)
95%	2,408 (59%)	1,736 (33%)	333 (40%)
99%	2,394 (59%)	1,693 (33%)	285 (34%)
100%	2,347 (58%)	1,141 (22%)	158 (19%)

3.4.3 Job scheduling

It is possible to establish an upper bound on the potential gain in power efficiency that can be achieved by power-aware job scheduling. To reduce the amount of infrastructure (PDUs, chillers, etc.) allocated to powering and cooling a supercomputer we wish to optimize the job schedule for the lowest *maximum* power draw. The same processes still have to run, so the overall energy is fixed. Hence, the best case is achieved by coscheduling high-power processes with low-power processes such that each supercomputer is running constantly at the mean power draw for the period (i.e., with no variability in power draw). The maximum potential reduction in peak power that can be attained by a perfect power-aware job scheduler (and assuming completely malleable applications) is therefore 1 – mean power/max. power. The data shown in Table 3.2 imply a maximum power improvement of 6.3% on Roadrunner, 29.8% on Cielo, and 52.1% on Luna. Again, this is a crude upper bound that makes some unrealistic assumptions about application characteristics, but it does indicate that there may exist the potential for a job scheduler to improve the power usage of a large-scale scientific workload, even ignoring possible savings from frequency and voltage scaling during application execution.

3.4.4 Energy usage

While most of our study focuses on power, energy—the integral of power over time—is also an important concern. We want to answer the following question:

> *For LANL's workload, if power is reduced to the bare minimum (i.e., idle power), how much slowdown in execution speed can the workload tolerate without increasing total energy?*

For example, if power were reduced by half, then any concomitant slowdown of less than 2x would result in a saving of energy while a slowdown of more than 2x would result in a squandering of energy.

We begin by determining the idle power for each of our three supercomputers. Figure 3.8 replots Figure 3.5 as a histogram, discarding outliers on the left part of Figure 3.8 that are presumably observed during full-system boot or power down. Note that each figure is plotted with different axes to clarify the shape of the curve; Roadrunner's horizontal range, for example, is extremely narrow. Idle power, drawn with a vertical red line in each histogram, was computed by first obtaining an estimate of the probability density function of power using a kernel density estimator [39]. The idle power was then selected as the lowest power that had a density value of at least 10% of the highest peak of the density curve. The use of a continuous density function makes this estimate of idle power independent of the bin widths displayed in Figure 3.8.

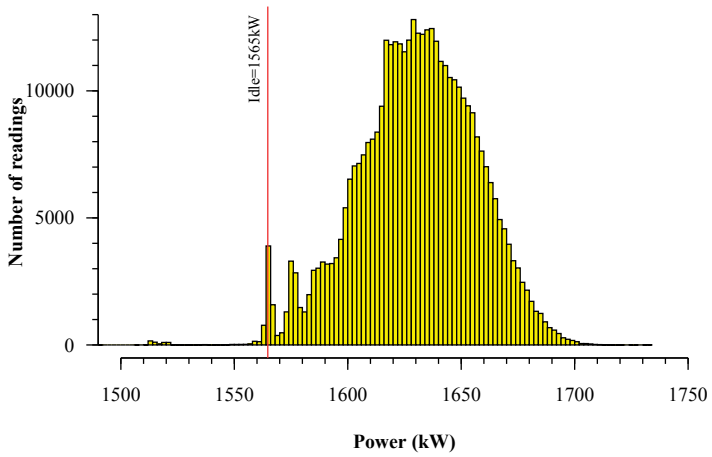
Given idle power, we can now treat that as the limit in power saving achievable by throttling processor frequencies and voltages. Table 3.4 presents the data and outcome of our energy calculation. By means of explanation, *Total time* represents Figure 3.5's horizontal range but with gaps elided; *Total energy* is the area under the curve in Figure 3.5; and *Idle power* is taken from Figure 3.8 as described above. *Scaled time*, the time that the entire workload would take if run at idle power but the same total energy, is computed as *Total energy* ÷ *Idle power*. Finally, *Max. energy-preserving slowdown*, is the maximum slowdown (i.e., factor increase in time) that can be observed without also increasing energy and is computed as the quotient of *Scaled time* and *Total time*.

Table 3.4: Maximal slowdown without increasing energy

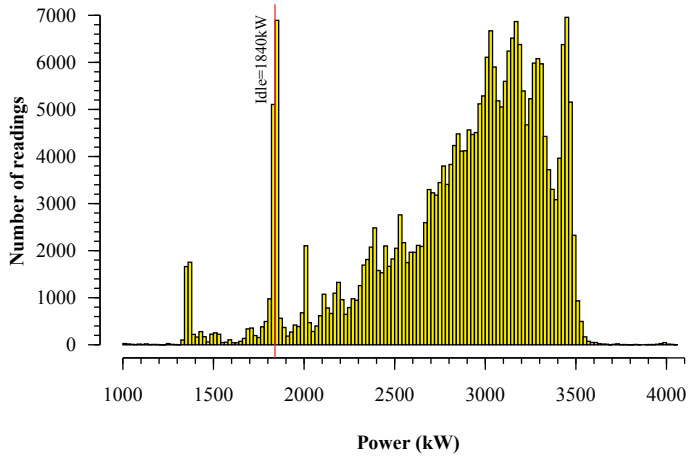
System	Total time (s) $= x$	Total energy (J) $= x \cdot y$	Idle power (W) $= y'$	Scaled time (s) $= (x \cdot y)/y'$	Max. energy-preserving slowdown $= y/y'$
Roadrunner	3.32×10^7	5.26×10^{13}	1.56×10^6	3.36×10^7	1.01 (1%)
Cielo	3.33×10^7	8.02×10^{13}	1.84×10^6	4.36×10^7	1.31 (31%)
Luna	9.79×10^6	3.08×10^{12}	2.12×10^5	1.45×10^7	1.48 (48%)

Figure 3.9 attempts to explain these terms and calculations graphically. The left side of the figure indicates the measured data, with *Total time* being x seconds, *Power* being y watts, and *Total energy* being the product of these, $x \cdot y$ joules. Reducing measured power (y) to idle power (y') produces the image on the right side of the figure. For the total energy to remain constant at $x \cdot y$ joules, the time must increase from x to x' to accommodate the reduced power budget. If the workload is heavily CPU-bound it will be sensitive to reductions in power produced by throttling the CPU speed. Reducing power—and correspondingly, CPU speed—from y to y' may therefore in practice increase execution time beyond x', leading to an increase (penalty) in energy consumption relative to that in the left side of the figure. If, however, the workload is minimally CPU-bound it will be robust to lower CPU speeds. In this case, reducing power from y to y' may increase execution time to less than x', leading to a decrease (benefit) in energy consumption. Realistically, no work can be performed at idle power, which is the "absolute zero" of power consumption. However, we use this value in our discussion because it provides an important upper bound on the time dilations that preserve energy.

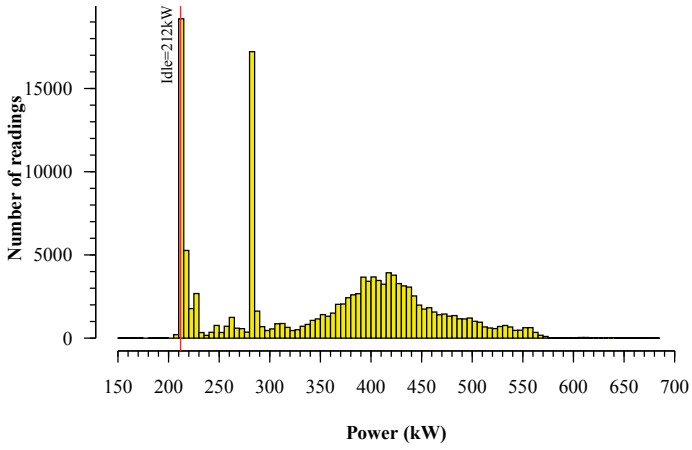
For example, consider a hypothetical supercomputer that draws 2 MW when idle plus an additional 4 MW when running the CPUs at full speed. An application that takes 100 s to run therefore consumes a maximum of 600 MJ. If the CPUs are downclocked to reduce the maximum CPU power to 3 MW (for a total system power of 5 MW), then the application will consume that same 600 MJ only if it can complete in 120 s.



(a) Roadrunner



(b) Cielo



(c) Luna

Figure 3.8: Histograms of supercomputer power usage over the time period from 1JAN2011 to 30APR2012

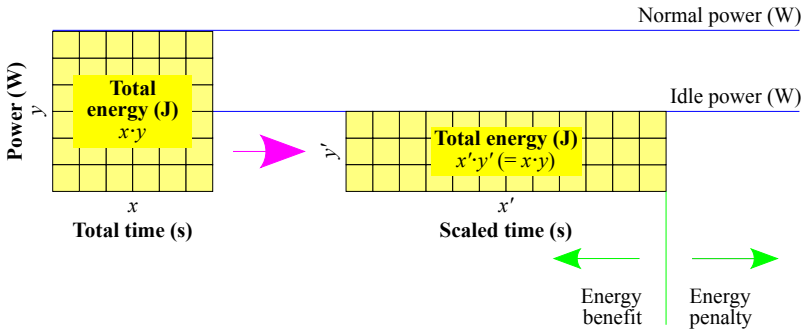


Figure 3.9: Explanation of slowdown computation performed in Table 3.4

This is possible only if the application is no more than 67% CPU-bound.[2] If the CPUs are downclocked to reduce the maximum CPU power to 0.5 MW (for a total of 2.5 MW), then the application will consume 600 MJ only if it can complete in 240 s. In the limit, where the CPUs are turned off (for a total system power equaling idle power, 2 MW), then the application will consume 600 MJ only if it can complete in 300 s. (Of course, only an application that spends all of its time blocked on memory, communication, file I/O, etc. can complete without using the CPU.) We therefore say that this hypothetical supercomputer's energy-preserving slowdown is (300 s − 100 s)/100 s or 200%.

Given this explanation, the conclusion one should draw from Table 3.4 is somewhat disappointing: Almost any slowdown in execution speed will result in an increased energy cost for performing the complete workload on any of the three supercomputers. Even Luna, the most tolerant of slowdown of the systems shown in the table, would need to run no more than 48% slower at idle power (CPUs turned off) than at full power in order to observe a net benefit in energy expended. Unless applications spend a substantial amount of timed blocked, energy consumption will increase when running at slower clock speeds.

On a more optimistic note, there may in practice be substantial room for lowering clock speeds without impacting performance. The performance robustness observed by Laros et al. [26] indicates that many applications *do* spend a lot of time with the CPU largely idle. Furthermore, many of the applications that run on Roadrunner, Cielo, and Luna spend a substantial amount of time in defensive I/O (checkpointing) [27] and blocked on network communication, during both of which the CPU is largely idle. Consequently, the results shown in Table 3.4 should be considered a worst-case (i.e., entirely CPU-bound) view of the improvements that can be achieved. The reality may be somewhat less pessimistic, but more research is needed to confirm this hypothesis.

3.5 Controlled studies

To complement our measurements of production workloads over a long time frame (Section 3.4) we additionally performed some controlled power studies on Luna. We were granted exclusive access to the entire Luna system for 10 hours and used this time in an attempt to answer the following questions:

[2] Non-CPU-bound codes are assumed to run at the same speed (1x) independent of power. CPU-bound codes in this example see no slowdown at normal power (6 MW) infinite slowdown at idle power (2 MW), and in general a slowdown of $4/(p-2)$ when given p MW of power. At 5 MW the slowdown for CPU-bound codes is therefore $4/(5-2) = 1.3$x. Hence, at 5 MW, for application time to increase from 100 s to only 120 s when fraction c of execution time takes 1.3x as long and fraction $(1-c)$ takes 1.0x as long, we solve $120\,\text{s} = 100\,\text{s} \cdot (1.3c + 1.0(1-c))$ and find that $c = 0.67$ or 67%.

- How well do full-system power readings at the switchboard match the aggregate intra-rack power readings?
- Are real applications' power characteristics qualitatively similar or different to those of kernel benchmarks?

Our methodology was as follows. We selected two kernel benchmarks to run—matrix-matrix multiplication (mmult) and the High-Performance LINPACK (HPL) benchmark [7]—and two representative LANL applications—xRAGE, a radiation-hydrodynamics code [16], and SPaSM, a molecular-dynamics code [40]. We began by letting Luna lie idle to get a consistent idle-power reading. Then we ran each of mmult, HPL, xRAGE, and SPaSM in turn, monitoring power at both the switchboard level and the "shelf" (10-node) level, allowing Luna to return to baseline power between runs. mmult is a single-process program and was run simultaneously on every core of the machine. The rest of the programs were run as 64-node jobs, filling the machine with those. We chose 64 nodes because, as Figure 3.10 indicates, over half of the jobs that run on Roadrunner, Cielo, and Luna utilize 64 or fewer nodes (Section 3.5), and over half of the elapsed wall-clock time on all three machines is spent running jobs comprising 64 or fewer nodes (Section 3.5). The cumulative density functions (CDFs) drawn in Figure 3.10 exclude single-node jobs, which are assumed to represent compilations, debugging sessions, interactive usage, and other tasks that are not production runs of scientific applications. However, they do include multi-node system health checks that are periodically enqueued through the regular queuing mechanism as there is no automatic way to elide that data.

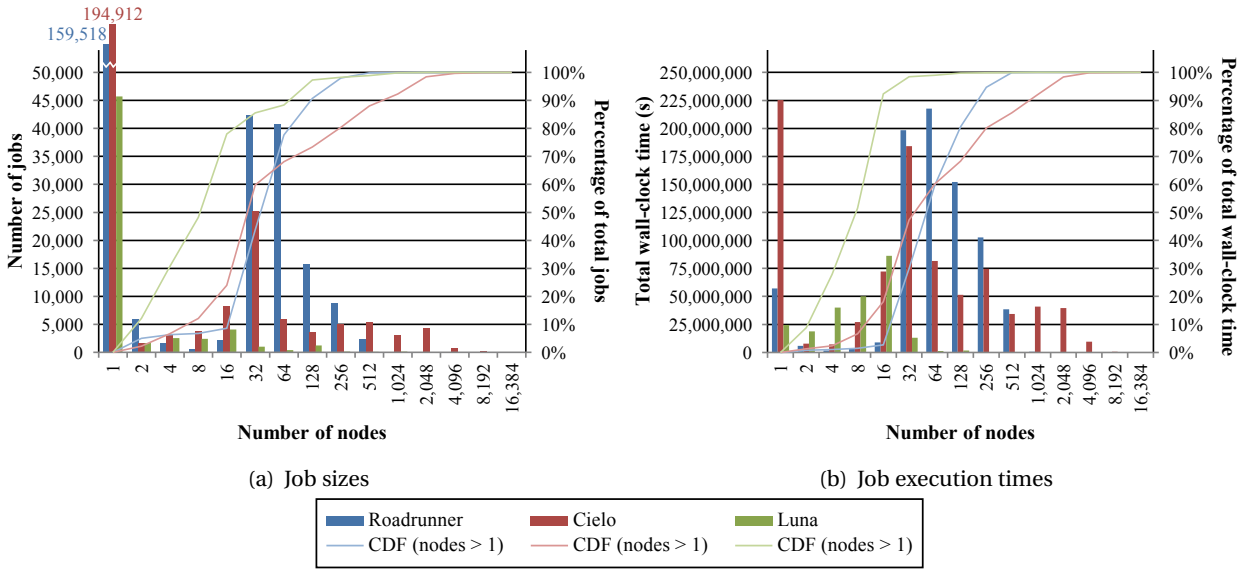


Figure 3.10: Histogram of job sizes and execution times over the period 1JAN2011 to 30APR2012

To ensure that our test workload does not leave the system in a different power state from how it began we then re-ran mmult and HPL to confirm that the power readings matched the previous readings. To quantify the impact of our choice of using 64-node jobs we ran a full-system (1,540-node) HPL job. Finally, after a long cooling-down period (partly including a few experiments that failed to launch), we re-ran SPaSM to get a second reading on its power usage.

When analyzing our data after our 10-hour session, we discovered, to our chagrin, that only 15 of the 154 shelf-level power monitors had returned reliable data; the rest returned zeroes for their power readings. Subsequent small-scale experimentation with Luna indicated that polling the power monitors

too quickly sometimes puts them into an error state. We therefore employed some statistical analysis of the good monitors to compensate for the missing data. As described below, this analysis indicates that even if all 154 power monitors had returned good data, our results would be unlikely to change by more than a few percent. The intuition behind this conclusion is that all of our benchmarks perform approximately the same work on all nodes. Hence, a power reading of any shelf is likely be representative of all of the other shelves.

A second challenge we encountered is that it was not possible to precisely synchronize in time the readings of the shelf power monitors (relative to the speed at which applications can change their power demands) so there is some skew in the readings. Again, we applied rigorous statistical analysis to adapt as much as possible to the available data.

Figure 3.11 plots the results of our experiments on Luna. In that figure, the x axes represent time of day (7:00 am to 5:00 pm), and the y axes represent either the total power drawn (Figures 3.11(a) and 3.11(b)) or the power difference between application-execution and idle times (Figure 3.11(c)). "HPL" represents the set of 64-node LINPACK jobs, and "HPL2" represents the full-system LINPACK job. Measurements of the 15 good shelf-level power monitors and the aggregate of the switchboard monitors are plotted in Figure 3.11(a). The thick, black line in the figure represents the switchboard power, and the the thin, colored lines represent per-shelf power. As Figure 3.11 indicates, the shelf power draws are fairly consistent. The shelf power draw curves were multiplied by 154 so that if every shelf behaved as that particular shelf, it would represent the cumulative power draw from all shelves. Also, these curves are not the raw data; a small amount of kernel smoothing was performed to fill in missing records. The switchboard power draw at idle, 212 kW, exactly matches the value computed in Figure 3.8(c), which lends credence to the statistical approach used in Figure 3.8 to empirically determine idle power.

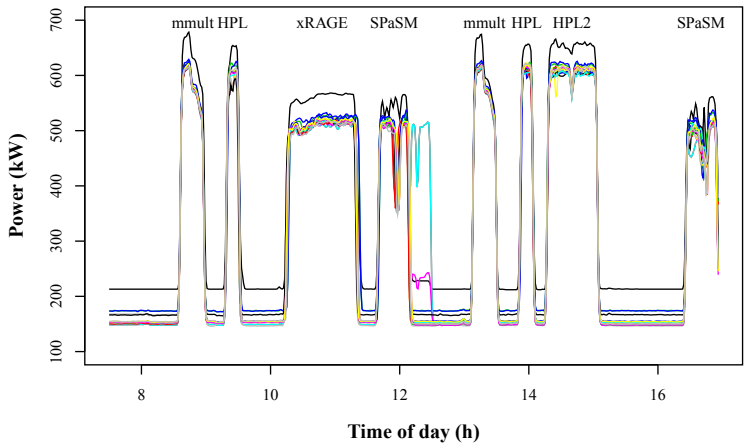
The dips in SPaSM's power draw are due to the SPaSM job in fact comprising two back-to-back launches of the application from within a single job script.

To get a handle on whether the differences between programs seen in Figure 3.11(a) are statistically real, we performed a one-way analysis of variance (ANOVA) on the maximum power draw during the execution of each code. This leads to largely significant differences between all programs except between HPL2 (the full-system LINPACK) and HPL (LINPACK on each node), and between HPL2 and mmult, both of which are mildly significant differences. SPaSM and xRAGE are also not significantly different in terms of expected maximum power draw.

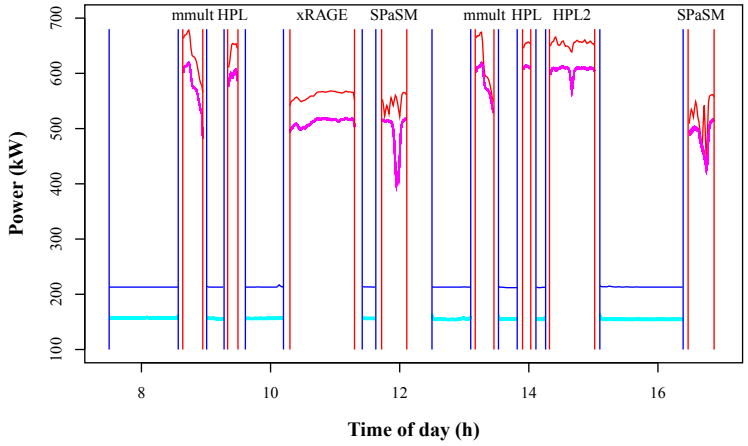
Because of time alignment issues—the shelf times lag by as much as 30 seconds—it is virtually impossible to recover any kind of useful comparison from switchboard to shelf power on ramp up and cool down (because they happen so rapidly). Therefore, Figure 3.11(b) plots the comparison between switchboard power draw and a projection of the cumulative shelf power draw during idle times (in between blue bars) and during the middle of code execution (in between red bars). The projection of cumulative shelf power draw was obtained by resampling the 15 shelf curves that were collected to fill in the "missing" 139 shelves to get plausible records for the total of 154 shelves. This can be considered an empirical Bayesian procedure [6] where the distribution of shelf power curves is estimated with the empirical distribution [38] (i.e., each observed curve is given probability 1/15), and the uncertainty in the total is then sampled according to the estimated distribution.

This process of sampling the missing shelf power curves, then adding them up produces many plausible curves for the cumulative shelf power draw. 1000 such curves are plotted in Figure 3.11(b) (in cyan for idle times and magenta for code execution). The band of cyan and magenta curves therefore includes the uncertainty in the cumulative shelf power resulting from our ability to obtain power data for only 15 of the shelves. However, this does *not* include uncertainty related to the time of measurement, which is likely important, especially for rapidly changing (in terms of power usage) codes like SPaSM.

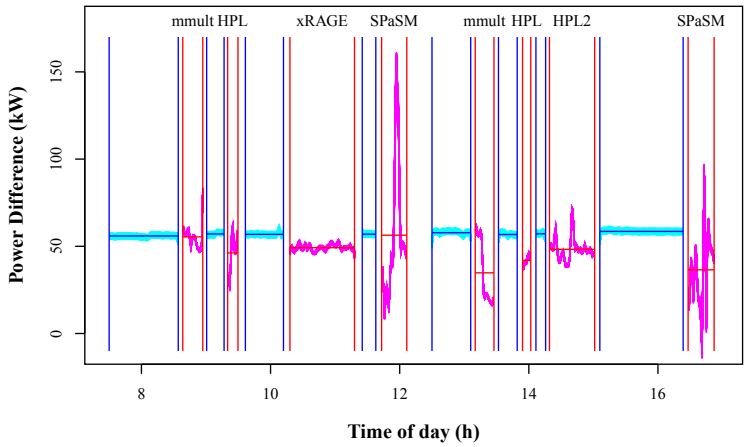
To highlight the differences between the switchboard curve and the plausible cumulative shelf power curves, Figure 3.11(c) replots the data as deltas between application-execution and idle times. Overall,



(a) Switchboard vs. shelf power



(b) Idle vs. busy power



(c) Idle vs. busy power deltas

Figure 3.11: Controlled power studies on Luna

there is about a 53 kW difference (averaged across all time in the plot and across plausible shelf power curves), with a 95% confidence interval for the average difference over the time of the experiment of (51.7, 54.4) kW (i.e., an accounting of the uncertainty in total shelf power). That is, if all 154 power monitors had returned nonzero readings, one could expect that the 53 kW value would not change by more than ±2 kW (±3.8%).

It is very difficult to assess the differences between switchboard and shelf power locally in time because of the time lag issues with the shelf-power readings mentioned above. This is particularly problematic for a job like SPaSM, whose power draw fluctuates rapidly during execution. Hence, it is unclear whether the differences between switchboard and shelf power between experimental conditions (including idle condition) in Figure 3.11(c) are due mostly to program differences or time resolution/accuracy issues. With that in mind, an ANOVA of the maximum power difference (between switchboard and shelf) during the 16 experimental trials (including the eight code runs and eight idle times as separate trials) found no significant differences between idle, mmult, HPL, xRAGE, and HPL2. However, SPaSM was significantly different from all other programs and from idle. Again, it is unclear how much of this is simply a time-alignment issue.

In our controlled study we considered both switchboard power and shelf power. However, recent processors such as the Intel Sandy Bridge used in Luna provide the ability to measure power consumption at both the socket and core level. We chose not to include such measurements in our Luna study because they perturb application performance. That is, a process must poll the on-chip power meters and log the results, and this introduces computational noise [33]. However, we did perform a separate study on a 150-node Luna-like system, Caddy. We ran a 2250-rank xRAGE job using only 15 cores per node while simultaneously running a power-monitoring program on each 16th core that logged power on a 10 s interval. The CPU cores, the "uncore" (everything on the socket except the CPU cores—caches, memory controllers, etc.), and other node hardware (everything in the node except the CPU sockets—memory, network cards, etc.) power are all highly correlated with shelf power. Specifically, we measured the power consumed at the shelf, core, and uncore level when running xRAGE at each clock frequency from 1.2–2.6 GHz in steps of 0.1 GHz. The shelf and core power measurements have a correlation coefficient of 0.99; the shelf and uncore power measurements have a correlation coefficient of 1.00; and the shelf and other-hardware (shelf/10 − (core + uncore)) have a correlation coefficient of 0.98. The implication is that shelf power provides an accurate representation of power usage without imposing a performance cost on applications (by reserving one core per node for monitoring or by introducing computational noise). However, the finer resolution provided by the on-chip power monitors does let us see, for example, that a node running xRAGE at the full 2.6 GHz spends an average of 48% of its power in the CPU cores, 53% in the rest of the socket, and only 9% in the rest of the node.

3.6 Future work

Having access to real-world power data on production supercomputers is an invaluable first step towards a variety of potential power studies. For starters, future work ought to correlate power data with job information. LANL maintains vast data on every job ever run on Roadrunner, Cielo, and Luna. Finding ways to link power measurements with job characteristics may result in some interesting insights regarding the way different applications consume power. If these insights lead to a predictive capability, then avenues for future research can include means for full-system power provisioning or power capping, broadening the node-level scope of current approaches such as Intel's Node Manager [19]. This can include job-scheduling aspects such as coscheduling high- and low-power jobs—or even phases within a job (e.g., computation versus file I/O)—to maintain a specified average power.

Given a system-wide power-capping infrastructure, an interesting follow-on study would be to quan-

tify the performance gains or losses incurred by running a larger number of nodes at lower power per node. That is, one could use the money saved by power capping (on both recurring costs such as energy and nonrecurring costs such as chillers) to purchase additional computer hardware (nodes, network switches, etc.). If the performace gains from increased parallelism offset the performance losses from power capping, then doing so will yield a benefit in computational throughput. This concept was in fact a large part of the motivation behind IBM's Blue Gene series of supercomputers [13]; power capping enables it to be applied to supercomputers in general.

While the xRAGE and SPaSM applications used in Section 3.5 are representative applications from the perspective of their patterns of computation, the normal LANL application workload comprises larger, longer-running applications; far more substantial problem sizes; and a significant amount of file I/O, especially defensive I/O (checkpointing) [27]. The implication of the last of those is that applications may have long, somewhat predictable phases of low power usage as large volumes of data are transferred from the supercomputer to a parallel filesystem and thereby represent an opportunity for power-aware coscheduling.

3.7 Conclusions

In this work we presented power measurements taken over a long period of time (16 months) on three large (Top100) and architecturally disparate supercomputers running production workloads. To our knowledge, this is the largest non-controlled study of power usage of a scientific workload performed at supercomputing scales. The following conclusions can be drawn from the data we presented in this paper:

- Variability in power draw can be quite different across different architectures, even when running a similar mix of applications.
- LANL's scientific workload draws substantially less power on average (70–75%) than the LINPACK benchmark. Consequently, if supercomputing facilities are specced to LINPACK's power needs, a substantial amount of trapped power capacity will be available to the data center.
- Throttling performance to maintain a maximum power level has the potential to succeed without disrupting the majority of applications. As an extreme example, Roadrunner can have its allocated power capped to only 41% of its existing value without impacting more than 1% of the workload that normally runs on that machine at LANL. Doing so would reduce the number of chillers needed, which, at a cost of US$580,000 per 1200-ton chiller, can represent a substantial savings in fixed costs. If power is further capped to reduce a supercomputer's actual power draw, our data show that the impact on applications may be small, but with a power and cooling cost of US$1 million per megawatt per year [11], this can translate to noticeable savings when aggregated across all of the supercomputers in the data center.
- A simple statistical analysis of our power data indicates that there may be opportunity for a power-aware job scheduler to reduce the LANL workload's peak power consumption by coscheduling jobs that consume little power alongside jobs that consume substantial power.
- There is so little difference between average and idle power on the three supercomputers studied that the best way to reduce the energy needed to perform a scientific workload is to run at full speed/maximum power rather than trying to reduce power and pay a penalty in execution time.

In summary, we have analyzed the power used by a production supercomputing environment. While our results are unlikely to precisely represent other supercomputing data centers, supercomputing platforms, or workloads, we believe that our methodology for analysis can be applied universally to determine salient characteristics about the systems in question and about the potential to achieve greater power efficiencies than what are currently observed.

CHAPTER 4

Power capping

The following text is copied essentially verbatim from the PADC project's statement of work.

The next step is to apply power capping techniques to existing systems to guarantee that the "worst case, feeder-sizing" demand does not occur, thereby liberating the trapped capacity to be used for additional systems. Power capping is typically implemented by using voltage and frequency controls to ensure that a system does not exceed a prescribed power demand limit.

Tasks

Most major system vendors have developed their own version of power capping. The lab that explores the power capping portion of this work shall perform the following tasks:

- Implement power capping and test on individual systems.
- Monitor systems to ensure that caps are not exceeded.
- Experiment with caps that are set below system peak demand levels.
- Characterize time/energy effects of power capped operation.

Milestones

1. Identify systems that have existing capping capability.
2. Implement capping of existing systems. Monitor and document results.
3. Document effects on execution time and total energy consumption of operating at a reduced power cap (below normal power demand).
4. Develop framework for managing power capping across heterogeneous systems within a data center.

4.1 Introduction

Section 3.4.2 provided evidence that there is opportunity on LANL's supercomputers for capping power with relatively little impact on LANL's production workload. To emphasize how great these savings can be, Figure 4.1 replots Table 3.3 as a bar chart. The x axis represents various power caps, not as absolute power levels but rather in terms of the fraction of data points in Figure 3.5 that would be impacted by the power cap. The data show that even without affecting a single data point, a power cap installed at the maximum power reading (the 0% mark on the x axis) can free up 22% of the power infrastructure allocated to Cielo, 58% of the power infrastructure allocated to Roadrunner, and 19% of the power infrastructure allocated to Luna. If the power cap were to be lowered to the point that 1% of the Figure 3.5 measurements would be affected, the potential savings in power infrastructure would increase to 33% on Cielo, 59% on Roadrunner, and 34% on Luna. Further lowering the power cap provides diminishing returns, although the power infrastructure allocated to Luna could be decreased by 50% of its original value when 25% of the data points lie above the power cap. Capping the power at the High-Performance LINPACK (HPL) level [7] offers relatively little savings because so few measurements in the actual workload are close to HPL measurements.

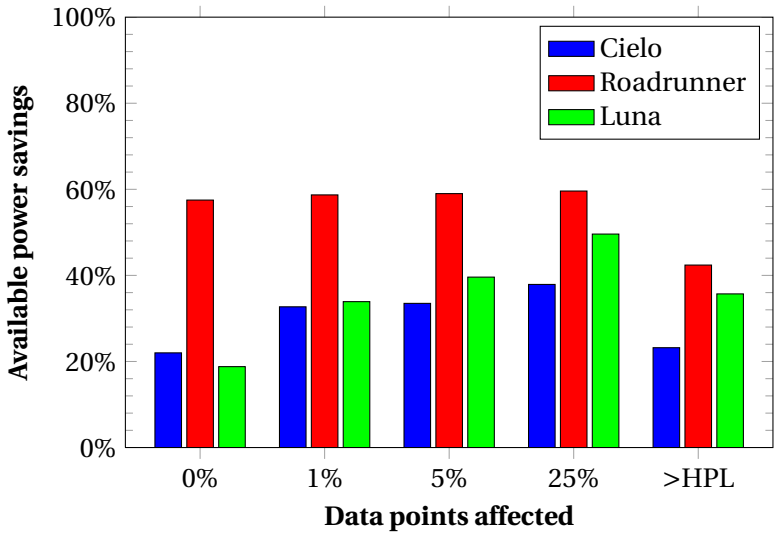


Figure 4.1: Potential savings from power capping

We developed our power-capping software on LANL's Caddy cluster. Caddy is a test-bed version of Luna. It is based on identical hardware [10] but at 1/10th the scale.

An important concern regarding power capping is the amount of performance that must be sacrificed to maintain a given power level. The switching power dissipated by a CPU built from static CMOS gates is described by $P = CV^2f$ [35]. However, out of the terms on the right-hand side—capacitance, voltage, and frequency—only frequency can be controlled directly from software. Hence, we consider how performance varies with respect to CPU frequency. Figure 4.2 presents some measurements we took of a single xRAGE [16] job running across all 150 nodes of Caddy as we varied the clock frequency from 1.2–2.6 GHz in steps of 0.1 GHz. The red data point represents the CPU's maximum clock frequency with Turbo Mode [15] enabled. Although this is nominally 2.601 GHz from the operating system's perspective, the actual clock speed is 3.0 GHz when all cores are active. Table 4.1 presents the Turbo-Mode clock frequency as a function of active cores for Caddy's (and Luna's) processors. The data were taken from the output of Intel's Power Thermal Utility (`ptumon`). Because the regression curve in Figure 4.2 has such a

tight fit ($R^2 = 0.9998$), one can accurately predict how long xRAGE would take to run at any of the clock speeds listed in Table 4.1 or even hypothesized.

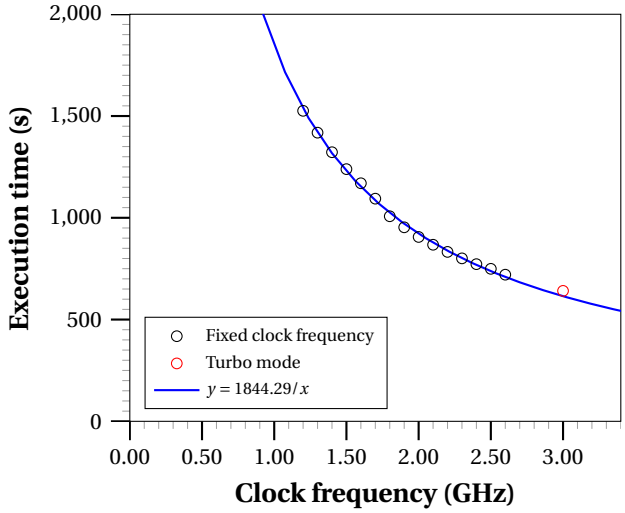


Figure 4.2: Impact of clock frequency on xRAGE performance

Table 4.1: Turbo Mode frequencies for the Intel Xeon E5-2670

Active cores	Clock freq. (GHz)
1	3.3
2	3.3
3	3.2
4	3.2
5	3.1
6	3.1
7	3.0
8	3.0

Figure 4.2 indicates that clock frequency largely determines application performance, especially for a compute-bound scientific code such as xRAGE. As the clock frequency increases by a factor of 2.5, the compute rate (i.e., the inverse of execution time) increases by a factor of 2.4. Any power-capping approach must respect that fact that reducing an application's power leads to a commensurate increase in execution time.

The rest of this chapter is organized as follows. Section 4.2 describes the approach we took to cap power at full-system rather than individual-node granularity. As our approach is based on a statistical model of power consumption over time, Section 4.3 presents and explains this model's derivation and use. Section 4.4 explains how we turned our power-capping ideas into a software implementation. Finally, Section 4.5 discusses where we left off at the end of the project and what work remains to be done.

4.2 Approach

Although power-capping hardware exists (e.g., Intel's Node Manager [19]), it runs at the scale of a single node. Our goal is to cap power on a full-system basis. A valid solution is to install power caps on all nodes such that the total system power lies beneath a given cap. However, this solution is suboptimal because a few nodes drawing much less power than the node-level cap (e.g., when blocked on I/O) cannot permit other nodes to run above their power cap, even though the total system may still lie beneath its power budget.

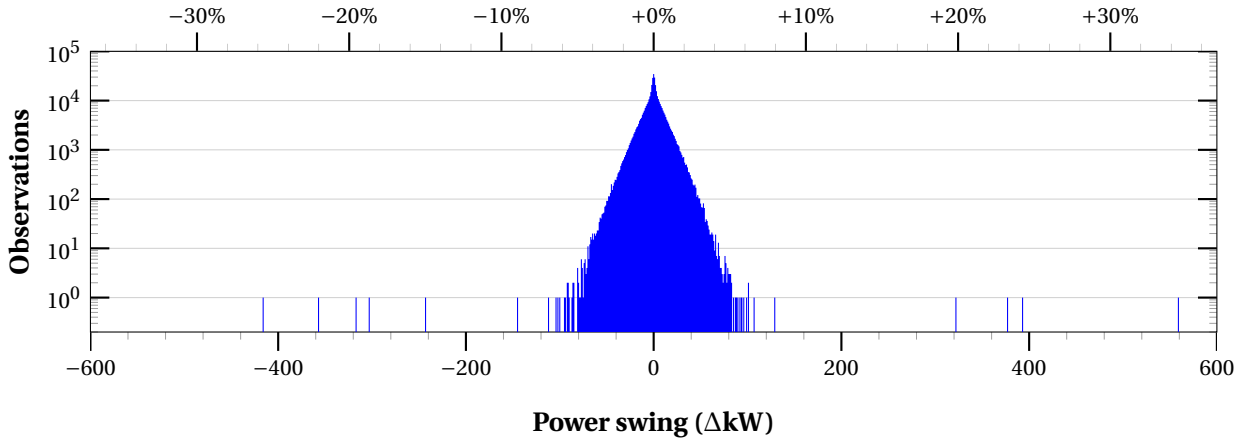
The major challenge with any software approach to power capping is that applications and even entire workloads can switch from drawing little power to substantial power faster than a software system can react, especially at scale. Figure 4.3 demonstrates the problem. The figure plots a histogram of power swings observed on Roadrunner, Cielo, and Luna within one-minute intervals, which is the rate at which we sample and log power data (cf. Section 3.3.1). The lower x axes represent the changes in each supercomputer's power consumption from one interval to the next. The upper x axes represent these changes as a fraction of the mean power consumption. The y axes—note the logarithmic scale—represent the number of times that that change in power was observed between 29FEB2012 and 19DEC2012 out of a total of over 400,000 valid measurements per supercomputer. Because the time range is different than that used in Section 3.4, the mean power consumption used to compute the upper x axes is also different from what is shown in Table 3.2 on page 13. For the range examined in Figure 4.3, we observed a mean power consumption of 1606 kW on Roadrunner, 2778 kW on Cielo, and 430 kW on Luna.

While the vast majority of the observations represent relatively little change in power, it is the outliers that raise concern. We turn our attention to Cielo, which draws the most power of the three supercomputers under consideration. There are 25 observations in Figure 4.3(b) in which power increases by more than 1 MW—36% of Cielo's mean power draw—within a single minute. At the extreme, Cielo once observed a sudden increase in power of 1.7 MW or 61% of its mean power draw. This represents an average increase of 28 kW per second. To put that rate in perspective, consider that four of the smaller machines on the November 2012 Top500 list draw about 40 kW when running LINPACK [29]. Cielo's largest power swing is therefore analogous to a small, Top500 supercomputer transitioning from being completely powered off to running LINPACK at full throttle within the span of about two seconds. It is virtually impossible for software to be sufficiently responsive to detect such rapid changes in power consumption across a supercomputer comprising many thousands of nodes, especially without also degrading application performance. If failure to enforce a power cap results in tripped circuit breakers, the consequences to users can be severe.

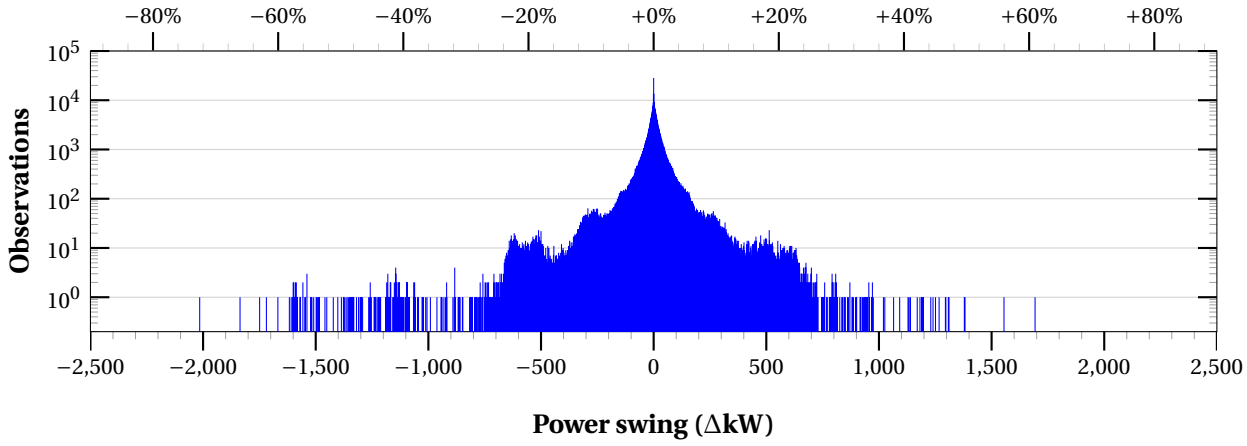
Although not as serious a concern, Figure 4.3(b) also indicates that there were 171 observations in which Cielo's power consumption *dropped* by more than 1 MW within a minute. At the extreme, an over-2 MW power drop (73% of mean power consumption) was observed within a single minute. This implies that software also has little time to detect that more of the power budget has become available and thereby grant applications more performance. As a result, application performance may be unnecessarily compromised.

Roadrunner draws fairly consistent power, with most points close to $x = 0$ in Figure 4.3(a). Luna is a much smaller system than Cielo yet observes power swings of similar relative magnitude as Cielo's. However, Luna observes large power swings more frequently, as indicated by the reduced weight around $x = 0$ in Figure 4.3(c) versus in Figure 4.3(b).

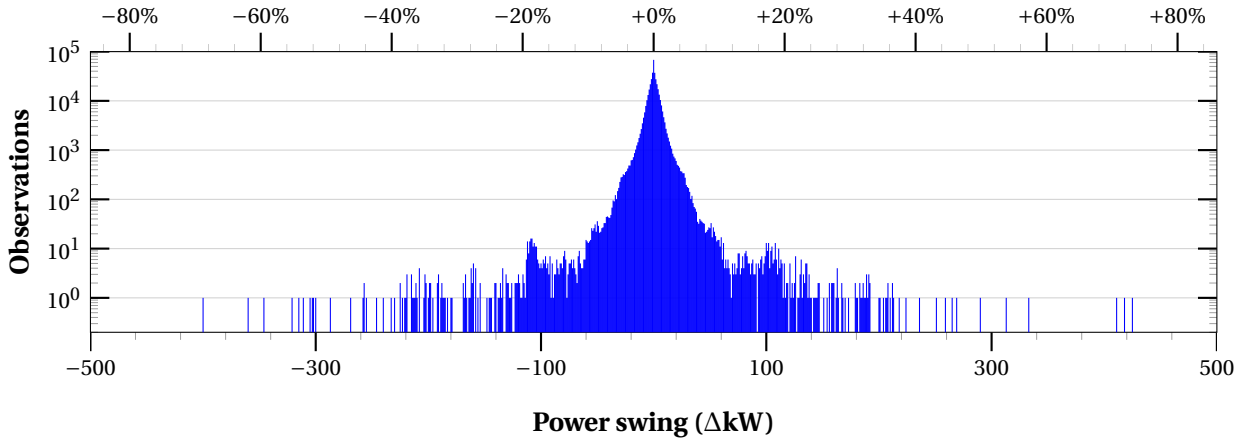
To address software's limited ability to respond to rapid increases or decreases in a supercomputer's power consumption our power-capping solution is based on a *statistical* approach. That is, we use past measurements of power consumption to predict how much power will be consumed in the near future and adjust processor frequency accordingly to enforce a power cap, albeit only within given statistical error bounds. The motivating intuition underlying this approach is that supercomputers such as LANL's observe



(a) Roadrunner



(b) Cielo



(c) Luna

Figure 4.3: Power swings observed within one-minute intervals (absolute and percent of mean)

a mixture of job sizes. A large job suddenly increasing its power consumption can lead to the power cap being exceeded; such jobs should have their processor frequency more conservatively controlled. In contrast, one can consider the likelihood of numerous small jobs suddenly and simultaneously increasing their power consumption to be a rare, "black swan" [41] event. Hence, small jobs can more liberally be granted higher processor frequencies with comparatively little risk of their exceeding a power cap.

4.3 Statistical model

Because software cannot respond fast enough to rapidly changing power conditions, the novelty of our power-capping approach is to rely on statistics to *predict* near-term power consumption. In this section we present more formally how our statistical model operates. Section 4.3.1 presents the model itself, and Section 4.3.2 demonstrates how the model is used by our power-capping software.

4.3.1 Statistical model for cluster power draw

The goal of the statistical model is to provide a risk quantification for the possibility of a large spike in power if we allow a certain job profile to run on a cluster. In this phase, usage data and power draw data for a given cluster are used to estimate the conditional distribution of power draw for a cluster given a particular job profile.

Let $y(t)$ be the total (switchboard-level) power draw for the cluster at time t. Predicting $y(t)$ is a challenge for a couple of reasons. First, $y(t)$ is a continuous time dependent process, and it is impossible to ignore this aspect and simply model $y(t)$ as independent over time (i.e., white noise). If $y(t)$ were a white noise process then, according to the model, $y(t)$ would most certainly spike above any finite bound in any positive length time, which is obviously not realistic or useful. However, a time dependent approach would require $y(t)$ to be similar to its recent values $y(r)$, $y(s)$, for $r < s < t$, in a manner that results in a continuous $y(t)$ curve. For example, one would be less surprised if a 650 kW power draw increased to 655 kW in a very short time while running the same jobs, than if a 200 kW power draw suddenly increased to 655 kW. The second challenge is that the variability (in addition to the mean) of the $y(t)$ process is dependent on the current job profile. We introduce a time series model that tackles these challenges below, and then discuss possible extensions in a future work section.

Once a model for $y(t)$ is in place, then a strategy can be implemented to select a job profile over time so that it is optimal in some manner (e.g., maximum throughput or least slowdown for all users, etc.) while being restricted so that the job profile at any given time has a very small chance (e.g., 1 chance in 1,000) of exceeding a set power threshold W. More specifics on this concept will be provided below after first discussing the estimation of the conditional distribution of $y(t)$ given the job profile.

Modeling Power Draw without power capping

For ease of presentation, we first consider the case where the CPU speed is 100% for all jobs in a given profile, and then consider the effect of lowering this later. Let $n(t)$ be the number of jobs running at time t. Finally, let $\boldsymbol{c}(t)$ be a vector of length $n(t)$ which contains the number of cores being used for each job at time t, and $\boldsymbol{s}(t)$ be the CPU speed allowed for each respective job in $\boldsymbol{c}(t)$. The goal is to characterize the distribution of $y(t)$ as a function of the job profile $\{\boldsymbol{c}(t), \boldsymbol{s}(t)\}$.

As discussed previously, a time dependent model must be employed for $y(t)$. The model used in this work is an order one auto-regressive (AR(1)) model [5], also known as an Ornstein–Uhlenbeck process [24] when used in continuous time. It is complicated by the following concerns: (1) the mean of this process is a function of the job profile, and (2) the distribution of the initial value of this process is a function of the job profile. We first consider the case where t is discrete (since the power data is recorded each minute)

for ease of presentation, but this will be generalized later. The full model can be written as follows for a given job profile with the convention that $t = 0$ is when this particular job profile was initiated.

$$y(t) = \mu + Z(t) \tag{4.1}$$

where μ is the conditional mean of the process, given the current job profile, and $Z(t)$ is the deviation off of the mean. The mean is assumed to take the form

$$\mu = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 \tag{4.2}$$

where x_1, x_2, and x_3, are predictors based on the job profile. Based on some preliminary model fitting, we have chosen to use $x_1 = n$, $x_2 = \sum c_j$, and $x_3 = \sum c_j^2$. In this case, the job profile $n(t) \equiv n, \boldsymbol{c}(t) \equiv \boldsymbol{c}$, is not dependent on time, since in Equation 4.1 we are modeling the power for a given profile, and therefore the profile remains constant over time. The parameters, α_1, α_2, and α_3, are unknowns that need to be estimated based on historical data. For Luna, the mean function μ in Equation 4.2 already accounts for 95% of the total variability in the power draw observed over the previous year. The $Z(t)$ process is essentially just a time series model of the residuals $y(t) - \mu$. Specifically, $Z(t)$ is an AR(1) process that can be written as

$$Z(t) = \begin{cases} Z_0 & \text{for } t = 0 \\ \beta Z(t-1) + \epsilon(t) & \text{for } t > 0 \end{cases}$$

where $Z_0 \sim N(0, \sigma_0^2)$, and $\epsilon(t) \sim N(0, \sigma^2)$, with σ_0^2 modeled as

$$\sigma_0^2 = \gamma_0 + \gamma_3 x_3, \tag{4.3}$$

and σ^2 modeled as

$$\sigma^2 = \delta_0 + \delta_3 x_3, \tag{4.4}$$

where γ_0,γ_3,δ_0, and δ_3 are restricted to be nonnegative. The variances σ_0^2 and σ^2 in Equations 4.3 and 4.4 are allowed to depend only on $x_3 = \sum c_j^2$. This decision was based on preliminary model fitting, but it makes intuitive sense as well. Suppose each of $n(t)$ jobs are running using $c_1, c_2, \ldots, c_{n(t)}$ cores, respectively, and each core from job j contributes v_j of additional power to the cluster. Then each job $1, 2, \ldots, n(t)$ contributes $c_1 v_1, c_2 v_2, \ldots, c_{n(t)} v_{n(t)}$ of additional power to the cluster, respectively. Assuming each job has the same variance, the variance of the additional power contribution from all jobs running would be

$$\text{Var}(c_1 v_1 + c_2 v_2 + \cdots + c_{n(t)} v_{n(t)}) = \sum c_j^2 \, \text{Var}(v_1).$$

Hence, it is not surprising that the models for variance preferred to depend on $x_3 = \sum c_j^2$.

The parameters β, σ^2, γ_0, γ_3, δ_0, and δ_3, are unknown parameters requiring estimation. All told, there are 10 parameters $\boldsymbol{\theta} = [\alpha_0, \alpha_1, \alpha_2, \alpha_3, \beta, \sigma^2, \gamma_0, \gamma_3, \delta_0, \delta_3]'$ in the model that need to be estimated. The estimate of these parameters, $\hat{\boldsymbol{\theta}}$ is obtained via maximum likelihood from the historical data. These estimates can also be updated over time in a manner that is discussed in the further work section below.

Modeling power draw with power capping

Now suppose that we are allowed to alter the CPU speed for each job in the current profile. Here we consider only one simple option, namely, dropping the CPU speed from the max speed S to πS (for some proportion $0 \le \pi \le 1$) for all cores from all jobs in the current job profile, i.e., $\boldsymbol{s} = [\pi S, \ldots, \pi S]'$, so that a job profile can be described by $\{\boldsymbol{c}, \pi\}$. This would have the effect of reducing the additional power that is

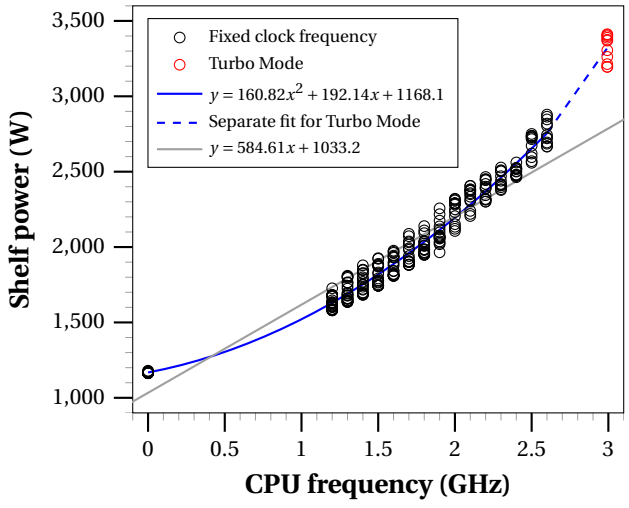
required above and beyond the baseline power draw. The baseline power is denoted by α_0 in Equation 4.2, as this would be the power drawn by the cluster if there were no jobs running (i.e., $n(t) = 0$). Combining Equations 4.1 and 4.2, we have

$$y(t) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + Z(t),$$

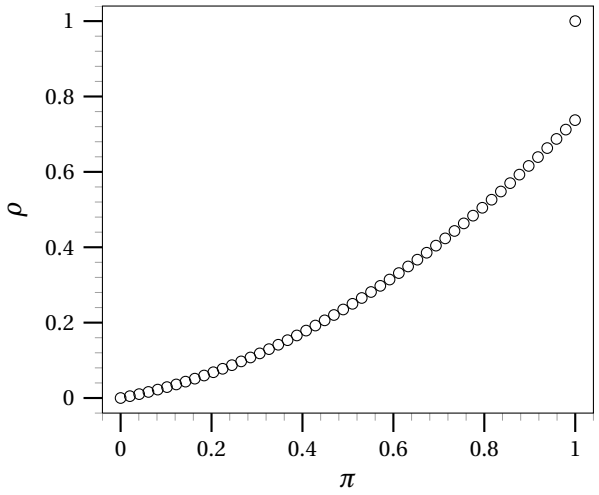
and suppose that the reduction in CPU speed to πS for all cores in use results in a power reduction of ρ for the additional power above baseline. This results in the expression

$$y(t) = \alpha_0 + \rho[\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + Z(t)]. \tag{4.5}$$

We only need to specify how ρ is determined from π. However, based on experimental data (obtained while running xRAGE [16] on Caddy), this relationship is well modeled by a quadratic until max speed, when it jumps into Turbo Mode [15]; see Figure 4.4(a), which presents power as a function of clock frequency when running xRAGE on Caddy. The blue line is the quadratic fit, and a linear fit is also included (in gray) to highlight the need for a quadratic. The red points are from running at maximum speed, which enables Turbo Mode. This jump is accounted for simply by treating Turbo Mode as a separate category in the regression.



(a) Experimental results



(b) Conversion from the regression line in (a) to the π and ρ relationship

Based on these data we assume that

$$\rho = \begin{cases} 1 & \text{if } \pi = 1 \\ \phi_1 \pi + \phi_2 \pi^2 & \text{if } \pi < 1 \end{cases} \tag{4.6}$$

where ϕ_1 and ϕ_2, are two more parameters that must be estimated from the data. This was accomplished in this case from the experimental runs on Caddy where π was varied while running xRAGE as seen in Figure 4.4(b). That figure shows the CPU frequency results converted from the regression line in Figure 4.4(a) to the π and ρ relationship, in this case, $[\phi_1, \phi_2] = [0.23222, 0.50556]$.

Ideally, these parameters would be estimated by running many different applications to ensure that the same relationship holds. If it does not, then it could be made a random effect, i.e., the parameters of the quadratic relationship are randomly chosen for a new application, to account for the additional variability. With some abuse of notation, group ϕ_1, and ϕ_2 in with the rest of the unknown parameters so

that $\boldsymbol{\theta} = [\alpha_0, \alpha_1, \alpha_2, \alpha_3, \beta, \sigma^2, \gamma_0, \gamma_1, \gamma_2, \gamma_3, \phi_1, \phi_2]'$, and again denote the maximum likelihood estimate of $\boldsymbol{\theta}$ as $\hat{\boldsymbol{\theta}}$.

4.3.2 Using the statistical model to characterize power

Once the parameter estimate $\hat{\boldsymbol{\theta}}$ is obtained, one can consider the following question: If a particular job profile, $\{\boldsymbol{c}, \pi\}$ were to run, what is the probability, $p(\boldsymbol{c}, \pi, W, T; \hat{\boldsymbol{\theta}})$, that it would exceed the power threshold W, in the next time horizon $T = 1$ hour (or 1 day, ...)? We would like to control the job profile so that $p(\boldsymbol{c}, \pi, W, T; \hat{\boldsymbol{\theta}}) < \eta$, where η is some small number (e.g., 1/1000). Assuming we have a way to evaluate the function $p(\boldsymbol{c}, \pi, W, T; \hat{\boldsymbol{\theta}})$, this can be accomplished by simply not allowing a job to start, or dropping the allocated CPU speed for one or more jobs until $p(\boldsymbol{c}, \pi, W, T; \hat{\boldsymbol{\theta}}) < \eta$. Here we consider only one option, namely, dropping the CPU speed for all cores from all jobs from the max speed S to πS for some proportion $0 \le \pi \le 1$.

Evaluating the Exceedence Probability for a Given Job Profile

The function $p(\boldsymbol{c}, \pi, W, T; \hat{\boldsymbol{\theta}})$ is difficult to evaluate in closed form for the model described in Section 4.3.1. However, using the model in Equation 4.5, many realizations (e.g., 10,000) of $y(t)$ can be made very quickly, and $p(\boldsymbol{c}, \pi, W, T; \hat{\boldsymbol{\theta}})$ can be evaluated by simply considering the empirical distribution function of the many realizations. Specifically, this works as follows.

For a dense time grid $\boldsymbol{t} = [t_0 = 0, t_1, \ldots, T]$, we use the statistical model in Equation 4.5 to generate N (with $N = 10,000$ for example) possible realizations of power for the given job profile $\{\boldsymbol{c}, \pi\}$ using algorithm 1 below. Denote the i-th of these realizations as $\boldsymbol{y}_i = [y_i(0), y_i(t_1), \ldots, y_i(T)]'$, $i = 1, \ldots, N$. Since we are interested in the probability that this profile could exceed the power threshold W in the next T time units, we can simply look at how many of the N curves exceeded W and divide by N. Namely,

$$p(\boldsymbol{c}, \pi, W, T; \hat{\boldsymbol{\theta}}) \approx \frac{1}{N} \sum_{i=1}^{N} I(\max(\boldsymbol{y}_i) > W),$$

where $I(A)$ is the indicator function for the event A, i.e., $I(A) = 1$ if A happens and 0 otherwise.

The algorithm for generating the realizations $\boldsymbol{y}_i$ for a particular profile is as shown in Algorithm 1. An example of using this procedure on Luna for a particular job profile is provided in Figure 4.4. Here 10,000 realizations were generated, but only 25 realizations are displayed for better visualization. Figure 4.4(a) displays 25 realizations of the job profile $\{\boldsymbol{c} = [20, 20, 200, 1300]', \pi = 1.0\}$. The entire 10,000 realizations were then used to estimate $p(\boldsymbol{c}, \pi, W, T; \hat{\boldsymbol{\theta}})$ for $W = 700$ kW, $T = 60$ min. In this case $p(\boldsymbol{c}, \pi = 1, W, T; \hat{\boldsymbol{\theta}}) = 0.268$, which is way too high of a risk. If we ramp down the CPU for all jobs by 1% (basically just taking all cores out of Turbo Mode), then we can obtain a corresponding set of 10,000 realizations, 25 of which are provided in Figure 4.4(b). Now $p(\boldsymbol{c}, \pi = 0.99, W, T; \hat{\boldsymbol{\theta}}) = 0.000$, and there is almost no risk in going over the cap with this job profile.

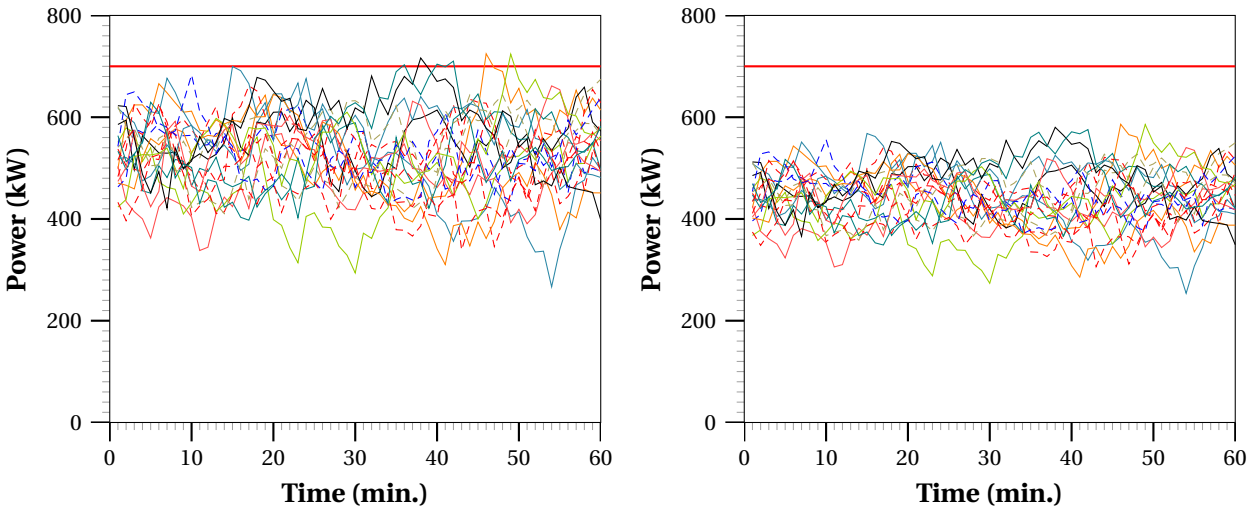
Algorithm 1 Generate a random power over time curve for a given job profile using the model in Equation 4.5

1: Calculate μ, σ_0^2, and ρ according to Equations 4.2, 4.3 and 4.6.
2: Convert the discrete time AR(1) parameters to those for a continuous time Ornstein–Uhlenbeck process (as the time grid $\boldsymbol{t}$ will not necessarily be at the same resolution of the data observations) using the relationship

$$\begin{aligned} \lambda &= -\log(\beta) \\ \tau^2 &= \sigma^2(-2\log(\beta))/(1-\beta^2) \end{aligned}$$

3: Generate $Z(t)$, the deviation from the mean:
4: Generate the initial value $Z(t_0) = \sigma_0 \xi_0$, where ξ_0 is a standard normal deviate.
5: **for** $j = 1$ **to** length($\boldsymbol{t}$) **do**
6: $\nu_j \leftarrow Z(t_{j-1}) \exp[-\lambda(t_j - t_{j-1})]$
7: $\omega_j^2 \leftarrow \frac{\tau^2}{2\lambda}(1 - \exp[-2\lambda(t_j - t_{j-1})])$
8: $Z(t_j) \leftarrow \nu_j + \omega_j \xi_j$
9: ...where ξ_j are independent standard normal deviates.
10: **end for**
11: Calculate power for each t_j according to Equation 4.5,

$$y(t_j) = \alpha_0 + \rho[\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + Z(t_j)].$$



(a) $\pi = 1.0$; probability of exceeding 700 kW is 0.268

(b) $\pi = 0.99$; probability of exceeding 700 kW is 0.000

Figure 4.4: 25 power realizations for 60 minutes for a job profile with $\boldsymbol{c} = [20, 20, 200, 1300]'$

4.4 Implementation

Our proof-of-concept implementation of power capping consists of a number of command-line programs and Unix dæmons that run at various locations in a cluster and that communicate with each other:

- The **state-change notifier** (Section 4.4.1) detects when a job is entering or leaving the cluster and notifies the master dæmon.
- When the **master dæmon** (Section 4.4.2) detects a change in cluster state (either periodically based on time or at job launch/termination from the state-change notifier) it determines what CPU frequency each job should utilize and notifies the node dæmons to adjust CPU frequencies accordingly.
- One **node dæmon** (Section 4.4.3) runs on each node of the cluster. The node dæmon's role is to set the CPU frequency of every processor on the node to a value specified by the master dæmon.

Figure 4.5 illustrates the juxtaposition of the various programs and what data are transferred from one to the next.

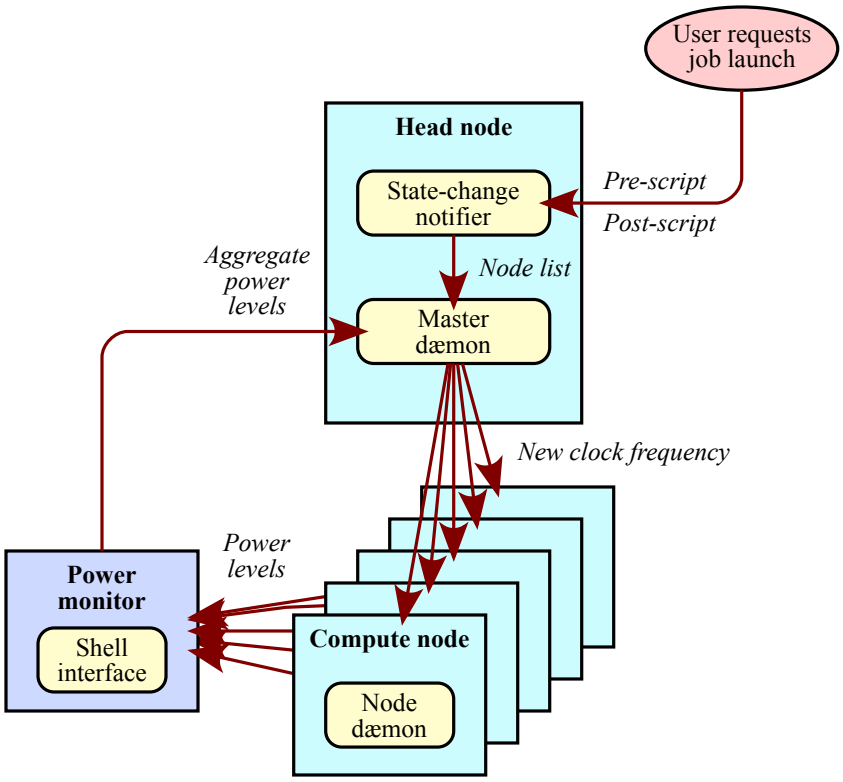


Figure 4.5: Components of our system-wide power-capping implementation

4.4.1 State-change notifier

The state-change notifier is a command-line program that the system administrator would normally incorporate into the SLURM prologue and epilogue scripts. The state-change notifier sends a message to the master dæmon containing (in ASCII) the word "BEGIN" (when the job is starting) or "END" (when the

job is terminating) followed by a space-separated list of host names. It outputs whatever text is returned to it from the master dæmon, which will be a success or error message.

4.4.2 Master dæmon

The master dæmon does most of the work for implementing power capping. It contains three main threads of control: one for communicating with the state-change notifier, one for communicating with the node dæmons, and one for communicating with the power monitors. The first thread receives notifications of jobs that are beginning or terminating, as described in Section 4.4.1. It updates the master dæmon's internal state to maintain a (bidirectional) mapping between nodes and jobs. These mappings are used to control power (via dynamic frequency scaling) on a per-job basis rather than a whole-system or per-node basis. The problem with modifying clock frequencies for the system as a whole is that this is too coarse of a knob to turn; aggregate power consumption would be able to increase or decrease only in crude steps, which would lead to excessive degradations in compute performance. The problem with modifying clock frequencies on a per-node basis is that LANL's supercomputing workload consists almost exclusively of bulk-synchronous [43] codes, in which applications alternate largely synchronized communication and computation phases. Because of this synchronicity, applications tend to run at the rate of their slowest process. Speeding up a subset of an application's processes will have little or no effect on the application as a whole. Slowing down a subset of an application's processes will slow down the entire application. The master dæmon therefore maintains the state necessary to control power at job granularity.

In addition to knowing what jobs are running where, the master dæmon needs to know how much power the system is drawing as a whole. It receives this information by periodically polling all of the shelf-level power monitors and summing the readings. In theory, we could read the switchboard-level power monitors, as there are fewer of them and they include power consumed by the network in addition to by the nodes. In practice, LANL's switchboard monitors reside in a different security domain from the supercomputers, and there is an air gap between the two, so this is not an option for us.

The master dæmon maintains some history of power usage to feed into the statistical model from Section 4.3 to predict future power usage. Whenever the master dæmon's view of system state has changed, either due to job entrance or egress or to a detected change in system-wide power consumption, the dæmon runs statistical simulations to determine the likely maximum power draw over the next unit of time. If this maximum exceeds the power cap specified by the system administrator, the dæmon makes a policy decision as to which nodes should have their CPU frequency lowered. If the maximum lies noticeably below the specified power cap, the dæmon makes a policy decision as to which nodes should have their CPU frequency raised.

Once the master dæmon has determined new CPU frequencies for each node, it notifies the node dæmons to effect the change.

4.4.3 Node dæmons

LANL's supercomputer usage model is such that nodes are not shared among jobs. Hence, it makes sense to adjust CPU frequencies for all cores on all sockets within a node rather than adjust on a per-socket (or per-core, if that were supported) basis. Our power-capping software therefore runs one node dæmon per node, and this dæmon is responsible for setting all CPUs in the node to a given frequency. As this is a privileged operation, node dæmons must be run with administrator privileges.

Node dæmons sit idle until they receive a message from the network. The message contains (in ASCII) the word "SETFREQ", followed by a frequency setting, followed by a list of nodes whose frequency should be set to the given value. The node dæmon bisects this list and sends a message listing half the nodes

to one of the nodes in the list and a message listing the other half to a node in that half. It then sets the frequencies of its own CPUs. This approach improves scalability as it enables the master to send a single frequency-change message per new, unique frequency setting to a single node dæmon before continuing with its other responsibilities. Meanwhile, the message is distributed to all N node dæmons in $\lceil \log_2 N \rceil$ steps, which even for a 10,000-node cluster represents only a few milliseconds of broadcast time.

4.5 Future work

While we made substantial progress in terms of understanding the issues involved in power capping, there are still many aspects of the topic that remain to be investigated. Section 4.5.1 discusses some of the policy decisions that need to be considered to determine what nodes should have their clock frequency altered as jobs enter and leave the system. Section 4.5.2 describes how the existing statistical model for predicting near-term power usage can be extended to take more aspects of the system into consideration and produce a more helpful prediction. And Section 4.5.3 discusses the engineering effort that needs to be exerted to transform our proof-of-concept power-capping implementation into something that can practically be deployed on a production supercomputer.

4.5.1 Policy decisions

Section 4.4.2 mentioned in passing that the master dæmon needs to make a policy decision whenever the system's potential power consumption changes: Which jobs should be allowed to run faster, and which should be required to run slower? In our proof-of-concept implementation we raise or lower all CPUs' clock frequency to the same level. Considering just the case of a new job entering the system, Table 4.2 lists some sample policies that the power-capping software might use to keep the cluster's total power consumption below the specified cap.

What should be clear from Table 4.2 is that there is not a unique, "right" approach. While all of the options presented can readily be implemented, the decision of which to employ is entirely a matter of policy that must be made by whoever manages the system.

4.5.2 Statistics

Updating of model parameters

In most practical cases, processes change over time. Sometimes this happens very gradually and sometimes more abruptly. In any case, this can be accounted for by allowing the model parameters $\boldsymbol{\theta}$ to change with time. For cluster power consumption, perhaps the applications that users are running evolve over time so that they become more or less power-hungry, or perhaps the variability in power consumption simply grows over time. However the parameter values change, this phenomenon is commonly treated with exponentially weighted methods such as exponentially weighted average (EWMA) [30] or exponentially weighted likelihood estimation.

A simple way to apply the EWMA strategy to this case is to estimate $\boldsymbol{\theta}$ weekly and combine the current week's estimate with previous weeks' estimates in a weighted average (so that the weights on each week decay exponentially). The parameter in the exponential decay is typically chosen to give the best out-of-sample prediction accuracy. The exponentially weighted likelihood works in a similar manner, but uses all of the recent data (e.g., last year's worth) in a maximum likelihood estimation that weights each data point differently (so that the weights on each data point decay exponentially in time). Either approach could be used effectively in this case. A more elaborate approach could allow the parameters of the time series model presented in Section 4.3.1 to also change according to their own time series

Table 4.2: Sample policies for reducing global power consumption

Policy	Pros	Cons
Slow down many small jobs	Small jobs are less likely to be performance-sensitive	May impact a large number of users
Slow down a few large jobs	A small per-job change can have a big, system-wide impact	Large jobs are often the most important ones
Slow down the jobs with the nearest completion time	Reduces the percent slowdown observed over a job's entire run	May frustrate users whose jobs were about to complete and suddenly are not
Slow down all jobs equally	Reduces per-job slowdown by amortizing across a large pool of jobs	May hurt performance more than necessary
Slow down the least compute-intensive jobs	Relatively little impact on performance	Relatively little reduction in system power consumption
Slow down randomly selected jobs	Fair across users and jobs	Introduces highly unpredictable and non-repeatable performance
Slow down jobs in decreasing priority order	Ensures highest priority jobs are the last to be slowed down	Requires a mechanism for associating priorities with jobs
Slow down the most recently launched jobs	Consistent performance for long-running jobs	Makes short jobs unbearably slow
Slow down the least recently launched jobs	Less likely for users to care about "slow" becoming "very slow"	Users annoyed by jobs not completed by expected deadline (e.g., when they arrive at work in the morning)
Delay the new job's launch	Applications see consistent performance across runs	Aggravates already-long waits for processor time

model. Such an approach would require advanced sequential estimation procedures such as sequential Monte Carlo [28]. In our experience with other problems like this one, however, the more mathematically advanced approaches such as this do not always improve performance over EWMA-type approaches.

Capping CPU by a different amount for each job

As discussed in Section 4.5.1, it may be advantageous in many circumstances to allow the cap on CPU speed to vary for the current jobs. Again, there are many possible ways to do this, and one may consider constructing some kind of optimality score $O(\{\boldsymbol{c}, \boldsymbol{\pi}\})$ for a particular job profile $\{\boldsymbol{c}, \boldsymbol{\pi}\}$, the form of which will depend on the policy decision. Note that π is now a vector of values for the proportion of maximum speed for each job. The goal, however, is always the same, namely to maximize O over $\boldsymbol{\pi}$ while keeping the risk of exceeding the power cap below the tolerance level η. There are many existing search routines that would be amenable to this case. However, the main concern becomes how to model the power in such a way that it can account for different changes in π for each job.

Such a model would need to account for the additional power draw (on top of baseline) due to each

job. A reasonable approach would be to use a construct similar to the discussion below Equation 4.4. That is, assume that each job's nodes increase the power draw of the cluster in lock step with one another. This node level increase can be modeled with a time series model similar to that in Section 4.3.1. Then, multiply the number of nodes in a job to the node level increases for that job to obtain the additional power draw due to that job. Finally, the total power is simply the baseline plus the sum of all of the power increases due to each job. This model is relatively easy to write down and should model the cluster power well based on our experience thus far. However, the main challenge for this model becomes the estimation of model parameters. In this model, each of the node level power increases are hidden processes in the sense that we only observe the aggregate of all jobs. This can create robustness issues for parameter estimation. One possible solution is to run experiments with only one job so that the increase due to a job is not hidden. This is costly and not always feasible however. An easier solution is to use penalized likelihood or Bayesian estimation approaches [14], which have been commonly used with success in such cases. Assuming a stable estimation routine can be put in place though, this model would provide exactly what is needed to predict power draw when varying π for each job.

4.5.3 Implementation

Our proof-of-concept implementation of power capping relies on some assumptions that are specific to the Caddy cluster:

- We assume that we can monitor power on "shelf" (10-node) granularity over a network via the Appro Greenblade 2 nodes' management and monitoring interface [3].
- We assume that SLURM [45] is used as the job scheduler.
- We assume that all of the nodes involved—the master node, the compute nodes, and the embedded power-monitoring interface—can communicate with each other over ordinary TCP [34].

None of these assumptions is fundamental to our design. Using different power-monitoring equipment or employing a different job scheduler would simply require rewriting the bits of code that interface to those components.

Our proof-of-concept implementation lacks a security model. In particular, SLURM's prologue and epilogue scripts, which we configure to invoke the state-change notifier, execute as the user who submitted the job, and all communication in our implementation is unauthenticated. Consequently, it is currently possible for a malicious user to trick the system into exceeding its power cap. We believe this shortcoming can be addressed in a relatively straightforward manner in a future version of the code.

Apart from the broadcast tree used by the node dæmons, code scalability has not been addressed in the proof-of-concept stage. As the number of power monitors increases, the effort needed by the master dæmon to query system power increases correspondingly. Splitting the single master dæmon into a hierarchy of dæmons would help distribute the load. Partitioning the cluster into separately power-capped sub-clusters would be another approach.

There is no fault tolerance or resiliency built into our design. If any dæmon fails, there is no ability to recover or work around the failure. A simple solution would be to install watchdog processes that monitor dæmons and restart them if necessary. Recovering from a failed master dæmon or node running the master dæmon is a trickier proposition, especially if the master dæmon is in fact replaced by a hierarchy of master dæmons as described above. Fortunately, this dæmon does not hold significant state. The mapping between nodes and jobs can be recreated by querying the job scheduler, and the current power state of each node can be determined by providing the node dæmons with a "GETFREQ" querying mechanism, presumably implemented as a reduction tree by analogy to the broadcast tree use

by "SETFREQ" messages. Both of these communication trees will require fault tolerance as well, to prevent a failed node dæmon from impeding message delivery to all downstream dæmons.

CHAPTER 5

Epilogue

In this final section of the report we first summarize our findings from the trapped-capacity and power-capping work (Section 5.1. We then draw some conclusions from those in Section 5.2. Finally, in Section 5.3 we describe some avenues for future research that follow nicely from the work done for the Power-Aware Data Center project.

5.1 Summary of findings

Los Alamos National Laboratory (LANL) was responsible for two key components of the Power-Aware Data Center (PADC) project: quantifying the amount of trapped power capacity in LANL's production supercomputing data centers and devising a scheme to exploit this trapped capacity by capping power, thereby freeing up power infrastructure for use by additional supercomputing capacity.

5.1.1 Trapped capacity

In Chapter 3 we analyzed the amount of trapped power capacity in three LANL supercomputers running their normal, production workloads: *Roadrunner*, an advanced architecture containing computational accelerators in every node; *Cielo*, an integrated, parallel supercomputer; and *Luna*, a commodity cluster. Our power measurements were performed at unprecedented scales:

- A year and a third of wall-clock time
- 10 MW of supercomputer power
- Three Top100 supercomputers totaling 3.2 Pflops/s—more supercomputing performance than exists in all but seven entire *countries*

Our measurements indicate that there is a substantial amount of trapped power capacity on Roadrunner, Cielo, and Luna. Table 5.1 recasts the data from Tables 3.1–3.3 to summarize the trapped capacity on each of the three supercomputers. In short, Table 5.1 indicates that LANL has a total of multiple megawatts of trapped capacity.

Our investigation into the sources of this trapped capacity led to an understanding of how power is allocated at LANL—and perhaps at other major supercomputing sites as well. It turns out that the data center's power infrastructure is upgraded and allocated to a new supercomputer far in advance of

Table 5.1: Summary of trapped power capacity

Trapped capacity relative to...	Roadrunner (kW)	Cielo (kW)	Luna (kW)
Maximum observation	2,347	1,141	158
LINPACK	1,735	1,204	300
Mean observation	2,457	2,345	513

supercomputer installation. In fact, the power budget is sometimes specified even before a supercomputer vendor is selected and the architectural details are confirmed. Consequently, nameplate power is little more than an educated guess as to a supercomputer's power requirements—and a liberal one at that to avoid procuring a supercomputer that cannot be powered on. Furthermore, power is never reallocated after a supercomputer is installed and actual power requirements are known because power preparations have already begun for the subsequent supercomputer.

5.1.2 Power capping

Although hardware solutions exist for enforcing a per-node power cap, implementing in software a power-capping scheme that caps power globally across a supercomputer's thousands of nodes and up faces some significant challenges. The most prominent of these is that a processor—and by extension an entire supercomputer—can ramp up its power consumption from the minimum to the maximum level on the order of milliseconds. This is extremely little time for software to detect and react, especially at scale and especially without having to run in an intrusive manner (e.g., by continuously polling power states on all processors).

The approach we take to overcome this challenge is to dynamically perform statistical analyses on historical and newly arriving power data. The key is to use a *time-dependent* model. A time-independent model would pessimistically observe that the maximum power draw, M, is observed every t seconds and would therefore predict that it will likely be observed again in the next t seconds. In contrast, our time-dependent model formally encapsulates the intuition that it is far less likely for supercomputer power to increase suddenly from $0.2M$ to M than it is for it to increase suddenly from $0.9M$ to M.

While more development and testing are required, we believe that this statistical approach is likely to be a sound way to establish a global power cap as long as it is tolerable for this cap to be exceeded on very rare occasions. For example, tripping a circuit breaker within a rack and causing a few nodes to reboot once every few months may be considered an acceptable risk. The algorithm does enable an administrator to trade off supercomputer performance for decreased likelihood of going over the power threshold, which is a benefit of our approach.

5.2 Conclusions

5.2.1 Trapped capacity

From the data presented in Chapter 3 we conclude that LANL's main supercomputing data center contains many megawatts of trapped power capacity. The exact amounts naturally depend on the precise definition of trapped capacity used (capacity over the maximum power ever observed, capacity over LINPACK power, capacity over some power quantile, etc.). However, even going by the most conservative estimates (maximum observed power), 58% of Roadrunner's power budget, 22% of Cielo's, and 19% of Luna's is trapped capacity. The reason these percentages are so high is that supercomputer power is budgeted long

before the supercomputer is installed to give the facilities engineers time to prepare the data center for the new procurement. Nameplate power therefore represents the supercomputer vendor's best estimate for the maximum power that the supercomputer will draw when running LINPACK. (This is what LANL specifies to the vendor instead of theoretical maximum power.) Because Roadrunner utilizes a novel, heterogeneous hardware architecture (Opteron and PowerXCell 8i [4]), LINPACK had not yet been ported or optimized to that platform at the time the data center had to be upgraded to support Roadrunner's power requirements.

Because LINPACK is used as LANL's power benchmark, this leads to another source of trapped capacity: Our data indicate that LINPACK is unrepresentative of LANL's scientific computing workload. According to our measurements, applications typically consume approximately 70–75% of LINPACK power.

In fact, such a small fraction of LANL's workload is power-hungry that power can be capped quite substantially with minimal impact on applications. For example, if Cielo's power were capped at 67% of its nameplate power, only 1% of the 278,394 power samples we observed over the course of 16 months would represent windows of time impacted by that cap.

While there exists a large gap between nameplate power and maximum power, there is relatively little difference between average power and idle power. If average power represents predominantly compute-bound application execution, then there may be little opportunity to save energy by reducing power, as execution time will increase at least at the rate that power is decreased.

Finally, we observed that different architectures see different variability in power draw and that power consumption can change greatly even over short time scales. Roadrunner sees very little variability because most of its power is consumed by the PowerXCell 8i accelerators, and these lack the sophisticated dynamic power adjustments of the Intel-architecture CPUs used in Cielo and Luna. In fact, even though LANL runs all of its supercomputers at maximum CPU frequency, disabling dynamic frequency scaling, our measurements indicate that at least on the Sandy Bridge processors (as used in Luna), dynamic voltage scaling does reduce the power consumed by idle processors regardless of CPU frequency.

5.2.2 Power capping

One conclusion from our power-capping effort is that a software approach to power capping is inherently limited due to software's slow response time. That is, software cannot practically throttle CPU frequency (the main controller available for power capping) fast enough to account for sudden increases in power. While hardware solutions exist for node-level power capping, there is no equivalent for capping power globally across a large supercomputer. Furthermore, it is questionable whether, at supercomputer scales, even hardware could successfully detect power swings and adjust CPU frequencies accordingly.

Per-node hardware power caps ought to be sufficient for throughput workloads, in which each node runs jobs that are independent of those on all other nodes. Alas, LANL's workload comprises tightly synchronized, parallel applications that run across many nodes. While per-node hardware power caps are sufficient in this case to prevent the power budget from being exceeded, they represent a suboptimal approach from a performance perspective. First, if parallel applications are tightly synchronized then they run only as fast as their slowest process. Hence, throttling the performance of one node belonging to each of two jobs is likely to hurt performance more than throttling the performance of two nodes of a single job. Second, it is acceptable for one node to run above the power cap as long as other nodes are running below the cap because this still maintains the global power cap. A node-level power-capping mechanism is too myopic to identify that situation and therefore may unduly compromise performance.

To overcome the limitations of slow response times we relied on statistical analyses to perform short-term predictions of how much power will be consumed globally based on prior measurements. We conclude that this is likely to be a practical approach to global power capping at supercomputer scales. By integrating with the job scheduler we provide the ability to raise or lower CPU frequencies on a

per-job rather than a per-processor, per-node, or per-system basis. This helps overcome the performance limitations of oblivious, per-node power capping implemented in hardware, as mentioned above.

Finally, as we implemented our global power-capping mechanisms we came to realize that there is a critical policy decision that must be made: Which jobs should have their CPU frequencies reduced to prevent the system from exceeding its global power cap? As we discussed in Chapter 4, there are many ways to answer that question, and each has its strengths and weaknesses. It is therefore important to realize that such policy questions exist and design a power-capping infrastructure to support as broad a set of policy decisions as possible, as the "right" decision cannot be known in advance and will likely be different across supercomputers, sites, and even time.

5.3 Future work

Our work on the Power-Aware Data Center project has opened up a few new avenues for follow-on research. We see the following as some intriguing approaches for gaining new insights into how supercomputer power usage can better be understood and optimized.

Combining hardware and software power-capping The advantage of establishing a power cap in hardware is that hardware can be more responsive than software to sudden changes in power consumption. However, software can better integrate with the job scheduler, which is needed to enforce policy decisions. We hypothesize that combining the two approaches will lead to a more robust, flexible solution to achieving power-capping on the scale of full supercomputer.

The idea is for software to monitor what jobs are running on the system and allocate power on a per-job basis according to the various policies put in place by an administrator. The software will then configure the per-node power-capping hardware to enforce those power budgets—with different caps on different nodes, based on workload an policy. As jobs enter and leave the system, the software will dynamically reconfigure the power-capping hardware accordingly to ensure that the global power cap is maintained while still granting performance where it is most needed.

Evaluating policy decisions There are myriad ways to select jobs whose performance should be throttled to meet a power cap. Section 4.5.1 enumerated a number of these and listed advantages and disadvantages of each. However, a question remains as to how a supercomputer administrator should select a policy given a known workload, set of users, and task priorities. If various policies could be applied faster than real time to historical job and power data, this could inform policy selection by identifying which jobs (and users) would have their performance throttled and by how much.

A useful comparison point would be to a static power scheme, for example one in which different machine partitions contain different numbers of nodes but the same total power budget. Figure 5.1 presents an example of such a configuration in which five partitions are each granted P power, but one partition comprises half the nodes in the cluster, one partition comprises a quarter of the nodes, and so forth. One advantage of such a structure over a policy-based power capping is that it enables users to reason about the power vs. performance trade off. That is, users running large jobs that spend a lot of time blocked on I/O can submit their jobs to a larger, lower-power partition while users running small, compute-intensive jobs can submit their jobs to a smaller, higher-power partition. Another advantage is that jobs have deterministic performance. Every time a particular job is run on a specific partition it will take roughly same amount of time to run; its performance will not depend on whatever else is running on the supercomputer at the same time.

Nodes	$\frac{N}{2}$	$\frac{N}{4}$	$\frac{N}{8}$	$\frac{N}{16}$	$\frac{N}{16}$
Power	P	P	P	P	P

Figure 5.1: Example of statically budgeting power by machine partition

Cost-oriented power usage The cost model used by power companies is complex and involves numerous factors. Base energy costs are pre-negotiated based on a band of expected power usage. In LANL's case, Los Alamos County—of which LANL is the major power consumer—negotiates peak and off-peak prices with the local power company for a given range of power. If the county uses less power than its range covers, it still pays the higher, pre-negotiated price. If the county uses more power than its range covers, its pays a substantially higher price, the exact amount depending on the amount of advanced notice. With sufficient advanced notice, the power company can spin up additional generators and deliver the power itself. Otherwise, power is delivered by another power company at a price determined by bid. In LANL's case the Dow Jones Four Corners spot-market price for electricity reflects prices set the preceding day for both peak and off-peak demand.

A question is whether this cost data can be utilized to make more cost-efficient use of a data center. If jobs can be moved from peak to off-peak hours on days when spot-market prices are high, this can result in large cost savings. Similarly, this cost data can feed into a power-capping system to adjust the cap based not just on the local power infrastructure (a hard limit) but also on energy prices at that time of day (a soft limit).

Modeling energy savings The energy model presented in Section 3.4.4 is naive in that it assumes that LANL's supercomputing workloads are completely compute-bound. Consequently, the conclusions drawn regarding the amount of slowdown that can be tolerated without increasing energy are extremely pessimistic. We believe that it is possible to develop an analytical model that accurately expresses how much energy can actually be saved on a real workload or given application through clock-frequency reductions. Doing so would be a useful contribution to the power-aware computing community as it can generalize all of the point solutions in the literature and explain power savings that others observe.

References

[1] Robert Alverson, Duncan Roweth, and Larry Kaplan. “The Gemini System Interconnect”. In *Proceedings of the IEEE 18th Annual Symposium on High Performance Interconnects*. (Mountain View, California, USA, August 18–19, 2010), pages 83–87. DOI: 10.1109/HOTI.2010.23.

[2] Kenneth Alvin, Njema Frazier, and Robert Meisner. *Advanced Simulation and Computing National Code Strategy: Simulation-Based Complex Transformation*. Publication NA-ASC-108R-09-Vol.1-Rev.0. Office of Advanced Simulation & Computing, NNSA Defense Programs, January 2009. URL: `http://nnsa.energy.gov/sites/default/files/nnsa/inlinefiles/ASC-Code-Strategy.pdf`.

[3] *Appro GreenBlade 2 SR5110–GB512X User Manual*. Version 1.0. Appro International, Inc. Santa Clara, California, USA, November 2011.

[4] Kevin Barker, Kei Davis, Adolfy Hoisie, Darren Kerbyson, Michael Lang, Scott Pakin, and José Carlos Sancho. “Entering the Petaflop Era: The Architecture and Performance of Roadrunner”. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*. (Austin, Texas, USA, November 15–21, 2008). ISBN: 978-1-4244-2835-9. DOI: 10.1109/SC.2008.5217926.

[5] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. 2nd edition. New York, New York, USA: Springer, March 8, 2002. 456 pages. ISBN: 978-0-387-95351-9.

[6] George Casella. “An Introduction to Empirical Bayes Data Analysis”. In *The American Statistician* 39(2) May 1985, pages 83–87. ISSN: 1537-2731. DOI: 10.2307/2682801.

[7] Jack J. Dongarra, Piotr Luszczek, and Antoine Petitet. “The LINPACK Benchmark: Past, Present and Future”. In *Concurrency and Computation: Practice and Experience* 15(9) August 10, 2003, pages 803–820. ISSN: 1532-0626. DOI: 10.1002/cpe.728.

[8] Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, Franck Cappello, Barbara Chapman, Xuebin Chi, Alok Choudhary, Sudip Dosanjh, Thom Dunning, Sandro Fiore, Al Geist, Bill Gropp, Robert Harrison, Mark Hereld, Michael Heroux, Adolfy Hoisie, Koh Hotta, Zhong Jin, Yutaka Ishikawa, Fred Johnson, Sanjay Kale, Richard Kenway, David Keyes, Bill Kramer, Jesus Labarta, Alain Lichnewsky, Thomas Lippert, Bob Lucas, Barney Maccabe, Satoshi Matsuoka, Paul Messina, Peter Michielse, Bernd Mohr, Matthias S. Mueller, Wolfgang E. Nagel, Hiroshi Nakashima, Michael E Papka, Dan Reed, Mitsuhisa Sato, Ed Seidel, John Shalf, David Skinner, Marc Snir, Thomas Sterling, Rick Stevens, Fred Streitz, Bob Sugar, Shinji Sumimoto, William Tang, John Taylor, Rajeev Thakur, Anne Trefethen, Mateo Valero, Aad van der Steen, Jeffrey Vetter, Peg Williams, Robert Wisniewski, and Kathy Yelick. “The International Exascale Software Project Roadmap”. In *International Journal of High Performance Computing Applications* 25(1) February 2011, pages 3–60. ISSN: 1094-3420. DOI: 10.1177/1094342010391989.

[9] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz André Barroso. "Power Provisioning for a Warehouse-sized Computer". In *Proceedings of the 34th Annual International Symposium on Computer Architecture*. (San Diego, California, USA, June 9–13, 2007), pages 13–23. ISBN: 978-1-59593-706-3. DOI: 10.1145/1250662.1250665.

[10] Michael Feldman. "Appro Comes Up Multi-Million Dollar Winner in HPC Procurement for NNSA". In *HPCwire* June 8, 2011. URL: http://www.hpcwire.com/hpcwire/2011-06-08/appro_comes_up_multi-million_dollar_winner_in_hpc_procurement_for_nnsa.html.

[11] Wu-chun Feng, Xizhou Feng, and Rong Ce. "Green Supercomputing Comes of Age". In *IT Professional* 10(1) January–February 2008, pages 17–23. ISSN: 1520-9202. DOI: 10.1109/MITP.2008.8.

[12] Wu-chun Feng and Heshan Lin. "The Green500 List: Year Two". In *Proceedings of the Sixth Workshop on High-Performance, Power-Aware Computing, 24th IEEE International Parallel and Distributed Processing Symposium*. (Atlanta, Georgia, USA, April 19–23, 2010). DOI: 10.1109/IPDPSW.2010.5470905.

[13] A. Gara, M. A. Blumrich, D. Chen, G. L.-T. Chiu, P. Coteus, M. E. Giampapa, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, T. A. Liebsch, M. Ohmacht, B. D. Steinmacher-Burow, T. Takken, and P. Vranas. "Overview of the Blue Gene/L System Architecture". In *IBM Journal of Research and Development* 49(2.3) March 2005, pages 195–212. ISSN: 0018-8646. DOI: 10.1147/rd.492.0195.

[14] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. 2nd edition. Chapman and Hall/CRC, 2004. ISBN: 978-1584883883.

[15] Paweł Gepner, Michał F. Kowalik, David L. Fraser, and Kazimierz Waćkowski. "Early Performance Evaluation of New Six-Core Intel Xeon 5600 Family Processors for HPC". In *Proceedings of the 9th International Symposium on Parallel and Distributed Computing*. (Istanbul, Turkey, July 7–9, 2012), pages 117–124. DOI: 10.1109/ISPDC.2010.23.

[16] Michael Gittings, Robert Weaver, Michael Clover, Thomas Betlach, Nelson Byrne, Robert Coker, Edward Dendy, Robert Hueckstaedt, Kim New, W. Rob Oakes, Dale Ranta1, and Ryan Stefan. "The RAGE Radiation-Hydrodynamic Code". In *Computational Science & Discovery* 1(1) October–December 2008. ISSN: 1749-4699. DOI: 10.1088/1749-4699/1/1/015005.

[17] Steve Greenberg, Evan Mills, Bill Tschudi, Peter Rumsey, and Bruce Myatt. "Best Practices for Data Centers: Lessons Learned from Benchmarking 22 Data Centers". In *Proceedings of the 2006 ACEEE Summer Study on Energy Efficiency in Buildings*. (Pacific Grove, California, USA, August 13–18, 2006). American Council for an Energy-Efficient Economy. URL: http://evanmills.lbl.gov/pubs/pdf/aceee-datacenters.pdf.

[18] P. Harvey, R. Mandrekar, Y. Zhou, J. Zheng, J. Maloney, S. Cain, K. Kawasaki, G. Lafontant, H. Noma, K. Imming, T. Plachy, and D. Questad. "Packaging the Cell Broadband Engine Microprocessor for Supercomputer Applications". In *Proceedings of the 58th Electronic Components and Technology Conference (ECTC)*. (Lake Buena Vista, Florida, USA, May 27–30, 2008). IEEE Press, pages 1368–1371. ISBN: 978-1-4244-2230-2. DOI: 10.1109/ECTC.2008.4550154.

[19] David Helman, Matthew Kelly, and Martin Guttmann. *Preserving Performance While Saving Power Using Intel Intelligent Power Node Manager and Intel Data Center Manager*. White Paper. Intel Corp., 2009. URL: http://www.intel.com/content/dam/doc/white-paper/intelligent-power-node-data-center-manager-paper.pdf.

[20] Bruce A. Hendrickson and David E. Womble. "The Torus-Wrap Mapping for Dense Matrix Calculations on Massively Parallel Computers". In *SIAM Journal on Scientific Computing* 15(5) September 1994, pages 1201–1226. ISSN: 1095-7197. DOI: 10.1137/0915074.

[21] Michael Hennecke, Wolfgang Frings, Willi Homberg, Anke Zitz, Michael Knobloch, and Hans Böttiger. "Measuring Power Consumption on IBM Blue Gene/P". In *Computer Science—Research and Developmen* 2011. ISSN: 1865-2034. DOI: 10.1007/s00450-011-0192-y.

[22] *InfiniBand Architecture Specification Release 1.2.1*. InfiniBand Trade Association. November 2007. URL: `http://www.infinibandta.org/specs/`.

[23] James A. Kahle, Michael N. Day, H. Peter Hofstee, Charles R. Johns, Theodore R. Maeurer, and David Shippy. "Introduction to the Cell Multiprocessor". In *IBM Journal of Research and Development* 49 issue 4/5 July–September 2005, pages 589–604. ISSN: 0018-8646. DOI: 10.1147/rd.494.0589.

[24] Olav Kallenberg. *Foundations of Modern Probability*. 2nd edition. New York, New York, USA: Springer, January 8, 2002. 648 pages. ISBN: 978-0-387-95313-7.

[25] Peter Kogge, Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Kerry Hill, Jon Hiller, Sherman Karp, Stephen Keckler, Dean Klein, Robert Lucas, Mark Richards, Al Scarpelli, Steven Scott, Allan Snavely, Thomas Sterling, R. Stanley, and Katherine Yelick. *ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems*. Technical report. Defense Advanced Research Projects Agency (DARPA), September 28, 2008. URL: `http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf`.

[26] James H. Laros III, Kevin T. Pedretti, Suzanne M. Kelly, Wei Shu, and Courtenay T. Vaughan. "Energy Based Performance Tuning for Large Scale High Performance Computing Systems". In *Proceedings of the 20th High Performance Computing Symposium*. (Orlando, Florida, USA, March 26–29, 2012). San Diego, California: Society for Computer Simulation International. ISBN: 978-1-61839-788-1. URL: `http://www.cs.sandia.gov/~jhlaros/publications/HPC2012_Laros.pdf`.

[27] Lawrence Livermore National Laboratory. *Attachment 2: Draft Statement of Work, Version 9. Advanced Simulation and Computing (ASCI) Purple*. Technical report UCRL-PROP-145639DR. Livermore, California, USA. Lawrence Livermore National Laboratory, February 15, 2002. URL: `http://www.llnl.gov/asci/purple/attachment_02_purplesowv09.pdf`.

[28] Jun S. Liu and Rong Chen. "Sequential Monte Carlo Methods for Dynamic Systems". In *Journal of the American Statistical Association* 93(443) September 1998, pages 1032–1044. ISSN: 1537-274X. DOI: 10.1080/01621459.1998.10473765.

[29] Hans Meuer, Erich Strohmaier, Jack Dongarra, and Horst Simon. *Top500 Supercomputer Sites: November 2012*. November 12, 2012. URL: `http://www.top500.org/lists/2012/12`.

[30] Douglas C. Montgomery. *Introduction to Statistical Quality Control*. 7th edition. Hoboken, New Jersey, USA: John Wiley & Sons, June 19, 2012. 754 pages. ISBN: 978-1-118-14681-1.

[31] National Nuclear Security Administration. *NNSA's Cielo Supercomputer Approved for Classified Operation*. Washington, DC, March 8, 2011. URL: `http://www.nnsa.energy.gov/print/mediaroom/pressreleases/cielo3811`.

[32] Venkatesh Pallipadi and Alexey Starikovskiy. "The Ondemand Governor: Past, Present, and Future". In *Proceedings of the Linux Symposium*. (Ottawa, Ontario, Canada, July 19–22, 2006). Volume 2, pages 215–230. URL: `http://www.linuxsymposium.org/2006/linuxsymposium_procv2.pdf`.

[33] Fabrizio Petrini, Darren J. Kerbyson, and Scott Pakin. "The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q". In *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing*. (Phoenix, Arizona, USA, November 15–21, 2003). DOI: 10.1109/SC.2003.10010.

[34] Jon Postel. *Transmission Control Protocol: DARPA Internet Program Protocol Specification*. Request for Comments 793. Marina del Rey, California, USA. Information Sciences Institute, University of Southern California, September 1981. URL: http://tools.ietf.org/pdf/rfc793.

[35] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits*. 2nd edition. Prentice Hall, 2003. 761 pages. ISBN: 978-0130909961.

[36] Andy Rawson, John Pflueger, and Tahir Cader. *The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE*. White Paper #6. The Green Grid, October 23, 2007. URL: http://www.thegreengrid.org/~/media/WhitePapers/White_Paper_6_-_PUE_and_DCiE_Eff_Metrics_30_December_2008.pdf.

[37] Bianca Schroeder and Garth A. Gibson. "A Large-Scale Study of Failures in High-Performance Computing Systems". In *IEEE Transactions on Dependable and Secure Computing* 7(4) October–December 2010, pages 337–351. ISSN: 1545-5971. DOI: 10.1109/TDSC.2009.4.

[38] Galen R. Shorack and Jon A. Wellner. *Empirical Processes with Applications to Statistics*. New York, New York, USA: Wiley, April 1986. ISBN: 978-0471867258.

[39] Bernard. W. Silverman. "Density Estimation for Statistics and Data Analysis". In *Monographs on Statistics and Applied Probability*. 1st edition. Volume 26. Chapman & Hall/CRC, April 1, 1986. ISBN: 978-0412246203.

[40] Sriram Swaminarayan, Kai Kadau, Timothy C. Germann, and Gordon C. Fossum. "369 Tflop/s Molecular Dynamics Simulations on the Roadrunner General-purpose Heterogeneous Supercomputer". In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*. (Austin, Texas, USA, November 15–21, 2008). IEEE Press. DOI: 10.1109/SC.2008.5214713.

[41] Nassim Nicholas Taleb. *The Black Swan: The Impact of the Highly Improbable*. 2nd edition. Random House Trade Paperbacks, May 11, 2010. ISBN: 978-0812973815.

[42] Patrick Thibodeau. "Los Alamos Shuts Down Supercomputers as Fire Advances". In *Computerworld* June 29, 2011. URL: http://www.computerworld.com/s/article/9218042/Los_Alamos_shuts_down_supercomputers_as_fire_advances_.

[43] Leslie G. Valiant. "A Bridging Model for Parallel Computation". In *Communications of the ACM* 33(8) August 1990, pages 101–111. ISSN: 0001-0782. DOI: 10.1145/79173.79181.

[44] Courtenay Vaughan, Mahesh Rajan, Richard Barrett, Doug Doerfler, and Kevin Pedretti. "Investigating the Impact of the Cielo Cray XE6 Architecture on Scientific Application Codes". In *Proceedings of the Workshop on Large-Scale Parallel Processing, 25th IEEE International Parallel and Distributed Processing Symposium*. (Anchorage, Alaska, USA, May 16–20, 2011), pages 1831–1837. DOI: 10.1109/IPDPS.2011.342.

[45] Andy B. Yoo, Morris A. Jette, and Mark Grondona. "SLURM: Simple Linux Utility for Resource Management". In *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing, 12th IEEE International Symposium on High-Performance Distributed Computing*. (Seattle, Washington, USA, June 24, 2003). Edited by Dror Feitelson, Larry Rudolph, and Uwe Schwiegelshohn. Volume 2862. Lecture Notes in Computer Science. Berlin/Heidelberg: Springer, 2003, pages 44–60. ISBN: 978-3-540-20405-3. DOI: 10.1007/10968987_3.

Contributors

The following people contributed to the work described in this report:

Robert E. Fields III
Hugh Greenberg
Gary Grider
Daryl Grunau
Craig Idler
Jeff Johnson
Michael Lang
Josip Loncaric
Ben McClelland
Sarah Michalak
Scott Pakin
Randal Rheinheimer
Eloy E. Romero, Jr.
Phil Sena
Curtis Storlie
Joanne Wendelberger